WebSphere® Adapters

Version 6 Release 2

WebSphere Adapter for Oracle E-Business Suite User Guide
Version 6 Release 2

WebSphere® Adapters

Version 6 Release 2

WebSphere Adapter for Oracle E-Business Suite User Guide Version 6 Release 2

Note

Before using this information and the product it supports, read the information in “Notices” on page 151.

December 2008

This edition applies to version 6, release 2, modification 0 of IBM WebSphere Adapter for Oracle E-Business Suite and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

Chapter 1. Overview of WebSphere Adapter for Oracle E-Business Suite

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

The adapter enables two-way communication between an application running on WebSphere Application Server 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 Oracle E-Business Suite 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 various versions of the Oracle database software; it supports the Oracle database server because the Oracle database server utilizes the Java™ Database Connectivity (JDBC) driver that supports the JDBC 2.0 or later specification. 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.

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 workarounds 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 Oracle E-Business Suite

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

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

An application running in WebSphere Application Server 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.

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.

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 Application Server.

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 object that contains the business objects changed by that event. The business object 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 style business objects. The adapter also supports both local and XA (distributed) transactions for outbound processing.

The adapter business object model uses the after-image style of business object for making updates. An *after-image* business object is one that contains the complete state of the business object after all desired changes have been made to it.

Supported operations

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

Table 1. Outbound operations supported by type of business objects

Business objects supported	Operation	After-image support
Tables Views Synonyms - Nick Names	Create	Yes
	Update	Yes
	Delete	Yes
	Retrieve	Not applicable
	RetrieveAll	Not applicable
	Exists	Not applicable
Stored procedures	Execute	Not applicable
Queries	RetrieveAll	Not applicable

Transaction management

The adapter supports both local and XA (distributed) transactions for outbound processing. In the adapter, a transaction is an isolated interaction with the database. A transaction can consist of multiple operations on the database that are performed as an atomic unit. These operations are not affected by simultaneously occurring operations from other users of the database.

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

- A *local transaction* is one in which a component defines the start and end of the transaction with a single database. It uses a one-phase commit protocol.

- An *XA transaction* is one in which the transaction can span multiple heterogeneous databases. It uses a global, or two-phase commit, protocol.

XA transactions

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

- Specify a JNDI data source that supports XA transactions, using the DataSourceJNDIName property
- Specify an XA data source, using the XADataSourceName property

The DataSourceJNDIName property represents a data source created within WebSphere Application Server. This name represents an XA or connection pool data source. If you define a JNDI data source that supports XA transactions on the server, and specify that data source when you configure the adapter, the adapter participates in XA transactions. Optionally, if you use an XA data source, the adapter participates in XA transactions.

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

   If the business object definition specifies that an attribute represents a child business object with single-cardinality and that attribute is empty, the adapter ignores the attribute. However, if the business object definition requires that the attribute represent a child, and it does not, the adapter returns an error and stops processing.
2. 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. If the retrieve operation is successful, the adapter continues the process of creating the parent business object; the adapter does not update the child business object without ownership.

   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.

3. Inserts the top-level business object in the database by performing the following actions:
   a. Sets each of the foreign-key values of the top-level business object to the primary key values of the corresponding child business object represented with single-cardinality. Because values in child business objects can be set by database sequences or counters or by the database itself during the creation of the child, this step ensures that the foreign-key values in the parent are correct before the adapter inserts the parent in the database.
   b. Generates a new, unique ID value for each attribute that is set automatically by the database. The name of the database sequence or counter is stored in the attribute's application-specific information. If an attribute has an associated database sequence or counter, the value generated by the adapter overwrites any value passed in by the application server.
   c. Inserts the top-level business object into the database.
4. Processes each of its multiple-cardinality child business objects as follows:
   a. Sets the foreign-key values in each child to reference the value in the corresponding primary key attributes in the parent. Because the parent's primary key values might have been generated during the creation of the parent, this ensures that the foreign-key values in each child are correct before the adapter inserts the child into the database.
   b. Inserts each of the multiple-cardinality child business objects into the database.

Retrieve operation:

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

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

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

   The Retrieve operation uses only the primary key. Other columns are ignored.
3. Recursively retrieves all multiple-cardinality child business objects.

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

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

Retrieving NULL data

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

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

RetrieveAll operation:

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

For database table business objects

All of the key and non-key attributes populated in the incoming business object determine the selection criteria for the retrieval. The adapter may retrieve multiple rows for the top-level business object from the database, depending on the attributes selected. 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 of the retrieved rows. The name of the container business object is the name of the business object with the string “Container” appended to it.

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`.
- `MatchesExceededLimitException` – The number of matching records in the database exceeds the value of the Maximum records for RetrieveAll operation property that is defined in the interaction specification. The MatchCount attribute of the fault contains the actual number of matches that the adapter found in the database, so that you can either increase the limit or refine the search.

Note: If the Maximum records property for RetrieveAll operation 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` – 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 J2C Bean wizard generates the query business object by running the user-specified SQL SELECT statement and creating a hierarchy of query business objects.

To process the query business object generated by the J2C Bean 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.
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 the entire 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:

In an Update operation, the source business object is compared to 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 the RecordNotFoundException exception, 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 ownership is true and 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.

   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 that is present in the source business object from the database. 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. If the retrieval fails, the adapter returns an ObjectNotFound exception.

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. If more than one row is returned, the adapter returns an error.

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 object 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
```

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.
- 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. The J2C Bean 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(x IN int, msgSTR INOUT VARCHAR(10), status OUT int,
                 outrec OUT $structname,  retArr OUT $arrayname)
```

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 J2C Bean wizard to discover table objects in a database, you can select multiple tables and add those tables to the selected objects portion of the Object Discovery and Selection screen. However, you cannot use the J2C Bean wizard to link or join the tables that you selected. If the goal of your business application will require 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 will vary 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` respectively.

   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 2 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**.
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 J2C Bean wizard to configure the adapter initially or at a later time by using the administrative console of the server to change the activation specification properties.

The database object that is the subject of the event is not retrieved until after notification is delivered to the export. As a result, the detection and notification of any retrieval errors that occur is deferred until after notification of the export. This differs from the event processing in version 6.0.2.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 the 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 Oracle_EBS_Adapter_Samples.zip to set up triggers for the adapter. The samples are located in the RAD_installation_dir/ResourceAdapters/ OracleApps_*version*/ directory.

The adapter offers assured once delivery, which guarantees that each event is delivered once and only once to the export. If you enable assured once delivery for the module, a transaction ID (XID) is set for each event in the event store. After an event is obtained for processing, the XID value for that event is updated in the event store. The event is then delivered to its corresponding export, and subsequently deleted from the event store. If the database connection is broken or the application is stopped before the event can be delivered, the event cannot be processed completely. In this case, the XID column indicates that the event must be reprocessed and sent to the export again. After the database connection is reestablished 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, time stamp, or connector id. The filter is set by use of the activation specification property EventTypeFilter, FilterFutureEvents, or AdapterInstanceEventFilter. The EventFilterType 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 and connector id. The adapter compares the system time in each poll cycle to the time stamp on each event. If an event is set to occur in the future, it will not be processed until that time. If the AdapterInstanceEventFilter activation specification property is set, only the connector id specified in the AdapterInstanceEventFilter property are processed.

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.

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

Custom event query

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

- Standard SQL statements
- A stored procedure
- A stored function

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

Standard SQL statements

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

Stored procedure

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

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

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

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

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

Stored function

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

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

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

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

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

Custom update and delete queries

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

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

Event store

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

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

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

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

Table 2. Definition of the event store database table

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

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

Column name	Type	Description
object_name	String	For runtime environments other than WebSphere Application Server: The name of the top-level Java bean that defines the business object. Each business object 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 event was generated. The format is `mm/dd/yyyy hh:mm:ss`.
event_status	Number	The event status. This is initially set to the value for a new event and updated by the adapter as it processes the event. The status can have one of the following values: • 0: Identifies a new event. • 1: Identifies an event that has been delivered to an export. • -1: An error occurred while processing the event. This column cannot be `null`.
event_comment	String	Any comment associated with the event.
connector_ID	String	The unique identifier for the adapter instance that will receive 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 Oracle E-Business Suite 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

Query business objects do not represent database objects. Query business objects represent a user-defined SQL query to run against the database.

Note: Before using the business objects to represent the above objects types, ensure that the Java keywords are not used to define the names of tables, views, stored procedures, and stored functions parameters.

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

A table business object always has a primary key, even if the corresponding database table does not have a primary key. The adapter uses the column specified 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 J2C Bean wizard prompts you for primary key information when you configure that business object. Specify a column that contains unique data, such as a sequence or identity column.

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

For stored procedure and stored function business objects

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

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

The Properties view below shows business objects 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 J2C Bean wizard generates a business object for the Struct type and adds it as a child object to the parent business object. 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.

In the Properties view, the ChildBOTypeName application-specific information is set to the value of the user-defined Struct type in the database.

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

How business objects are created

You create business objects by using the J2C Bean wizard, launched from Rational® Application Developer for WebSphere Software. 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.

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.

In some instances, the wizard cannot completely configure a parent-child relationship. For these relationships, you use the business objects editor, launched from Rational Application Developer for WebSphere Software, to modify or complete the definition of a business object hierarchy that was created by the wizard.

Business object hierarchies

Define the relationships between database tables using parent-child relationships and data ownership in hierarchical business objects.

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.

The term *hierarchical* business object refers to a complete business object, including all the child business objects that it contains at any level. The term *individual* business object refers to one business object, independent of child business objects that it might contain or parent business objects that contain it. The individual business object can represent a view that spans multiple database tables. The term *top-level* business object refers to the individual business object at the top of the hierarchy, which does not itself have a parent business object.

A hierarchical business object has attributes that represent a child business object, an array of child business objects, or a combination of the two. In turn, each child business object can contain a child business object or an array of child business objects, and so on.

A *single-cardinality relationship* occurs when an attribute in a parent business object represents one child business object. In this case, the attribute is of the same type as the child business object. The adapter supports single-cardinality relationships, and single-cardinality relationships and data without ownership.

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

Use the following types of relationships between business objects to define a hierarchy that represents your database tables:

- Single-cardinality relationships
- Single-cardinality relationships and data without ownership
- Multiple-cardinality relationships
- Child business objects with multiple parents

In 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.

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

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

Single-cardinality relationships and data without ownership

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

However, situations can occur in which multiple hierarchical business objects contain the same data, which none of them owns. For example, assume that the Address database table contains a reference to the StateProvince lookup table. Because the lookup table is rarely updated and is maintained independently of the address data, creating or modifying address data does not affect the state and province data in the lookup table. However, to be able to retrieve the StateProvince business object along with the Address business object, StateProvince must be a single-cardinality child of Address and the relationship must be defined without data ownership.

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

When the adapter receives a hierarchical business object with a Create, Delete, or Update request, the adapter does not create, delete, or update single-cardinality child business objects contained without ownership. The adapter performs only Retrieve operations on these business objects. If the adapter fails to retrieve such a single-cardinality business object, it returns an error and stops processing; it does not add or change values in the lookup table’s business object.

Denormalized data and data without ownership

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

Synchronization of normalized to denormalized data: When the relationship is without ownership, you can create or change data when you synchronize from a normalized application to a denormalized one. For example, assume that a normalized source application stores data in two tables, A and B. Assume further that the denormalized destination application stores all the data in one table such that each entity A redundantly stores B data.

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

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

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

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

Multiple-cardinality relationships:

In a multiple-cardinality relationship, an attribute in the parent business object represents an array of child business objects. The attribute is of the same type as the child business object. The foreign key that describes the relationship is stored in the child, except when an application stores a single-child entity. Then the parent-child relationship is stored in the parent.

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

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

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

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.

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 editor. The J2C Bean wizard configures only one parent.

The business object schema

The business object schema is built out of database objects that you select when you run the J2C Bean 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 OracleEBSASI.xsd. The schema file generated for the adapter has a reference to this template in its annotations.

Stored procedure overview

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

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

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

Stored procedure business object overview

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

The J2C Bean wizard helps you build stored procedure business objects that run a stored procedure or stored function. The wizard examines the stored procedure or stored function in the database to create the business object. A stored procedure business object has an attribute for each parameter.

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

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

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

During discovery and at run time, the WebSphere Adapter for Oracle E-Business Suite expects the returned ResultSets from the stored procedure execution to contain columns with names. Some stored procedures return ResultSets with unnamed columns. For example, a stored procedure with the SQL statements similar to the examples that follow will return ResultSets 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 ResultSet- like `count(*)` or `d1, d2, d3` for the respective select statement examples shown above.

If the returned ResultSet 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 of these nested Struct and Array objects.

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

Property name	Type	Description

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

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

Stored procedures used in place of or in addition to operations

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

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

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

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

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

Table 5 lists the application-specific information elements for a stored procedure and describes their purpose and use. A complete description of each element is provided in the sections that follow the table. A screen showing the stored procedure definition for a business object is shown in "View of business object with stored procedure definition" on page 28.

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

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

Stored procedure type

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

Table 6. Stored procedure type element characteristics

Required	Yes
Default	None

Table 6. Stored procedure type element characteristics (continued)

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

Stored procedure name

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

Table 7. Stored procedure name element characteristics

Required	Yes
Default	None
Bidirectional transformation supported	Yes
Property type	String

Result set

This value determines whether the stored procedure returns a result 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

Table 8. Result set element characteristics (continued)

Property type	Boolean
Usage notes	If your stored procedure returns a result set, use the business object editor after finishing the J2C Bean wizard to verify that this attribute is set to `true`. The Oracle JDBC driver does not always return this value correctly.

Parameters

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

Table 9. Parameters element characteristics

Required	Yes
Default	None
Contents	Each Parameters element specifies the following information: • PropertyName: Specifies the name of the business object attribute to pass as the parameter. • Type: Specifies the type of the parameter, one of the following values: – `IP` for input only – `OP` for output only – `IO` for input and output – `RS` for result set
Bidirectional transformation supported	No
Property type	String
Usage notes	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 a value is returned.

Table 10. Return value element characteristics

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

Table 10. Return value element characteristics (continued)

Usage notes	If the returned value is RS, the returned value is a result set and is used to create the multiple-cardinality container corresponding to this business object. If the returned value is the name of an attribute, the value is assigned to that particular attribute in the business object. If the attribute is another child business object, the adapter returns an error. When you associate a stored procedure with a business object that is generated from a table or view, and if the stored procedure is a function, a value will be returned from this stored procedure. One ReturnValue application-specific information value is added to the operation application-specific information. The existence of this application-specific information implies that it is a function call and not a procedure call, because a value is being retuned by the function. If the value of this application-specific information is a business object attribute name, the returned value is assigned to that particular attribute in the business object. If the value of this application-specific information is another child business object, the adapter runtime returns an error. In summary, if the returned value is of a simple data type, the wizard enables you to bind one business object attribute to it, and the value of this application-specific information is set to the name of that business object attribute. But if the returned value is a result set, the wizard sets the value of this application-specific information to RS. **Note:** A result set can be returned as an output parameter or as a returned value if it is a stored function. The type of the output parameter is set to RS to indicate that this parameter is used to return a result set.

View of business object with stored procedure definition

The following Properties view screen shows the customer business object that has the associated stored procedure information for RetrieveSP and AfterRetrieveSP for the Retrieve operation. The adapter runs the RTASSER.RETR_CUSTNAME stored procedure in place of the standard SQL to retrieve a table business object. After the business object is retrieved, the adapter runs the RTASSER.RETR_CUSTINFO stored procedure.

Stored functions overview

The Oracle database supports stored functions in addition to stored procedures. Stored functions are similar to stored procedures except that they always return a value. The adapter supports them in a similar manner.

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

A function call has the following syntax:

```
? = call FunctionName parameter_list
```

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

```
call SPName parameter_list
```

You specify the attribute that contains the returned value by using the ReturnValue business object application-specific information.

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 J2C Bean wizard helps you build query business objects that run user-defined SELECT statements against the database. You specify the SELECT statement, using ? (the question mark) in place of any substitutable parameters in the SELECT statement. The wizard then provides an area where you specify the data type of each parameter and provide a sample value. The sample value must match data in the database because wizard uses the SELECT statement's results to create the query business object.

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

For example, assume you specify the following SELECT statement:

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

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

Java data bindings

The business data exchanged between the client application and the resource adapter is represented as Java Data Bindings. The metadata describing the business data is defined as business objects and represented as the XSD schemas. The Java Data Bindings are generated from these XSDs and are the realization of the business objects.

A java data binding is a structure that consists of data and, in some cases, metadata with additional instructions, for processing the data. It is a generated, hierarchical, java objects implementing Record interface. The data can represent a business entity, such as an invoice or an employee record.

You create java data bindings by using the J2C Bean wizard, launched from Rational Application Developer Connector Tools. The wizard connects to the Oracle E-Business Suite system, discovers data structures in the EIS, and generates data bindings to represent them. The adapter supports records that are hierarchically structured. Information about the processed object is stored in the application-specific information for the object and each of its attributes.

The J2C Bean wizard

The J2C Bean wizard is a tool you use to build application interacting with the Oracle E-Business Suite system before deploying it to WebSphere Application Server. The wizard establishes a connection to the Oracle E-Business Suite server, discovers business objects and services (based on search criteria you provide), and generates Java data bindings based on the services or functions discovered.

Using Rational Application Developer for WebSphere Software you establish a connection to the Oracle E-Business Suite server to browse the metadata repository on the server. In the wizard, you select the business objects you need. The wizard automatically generates the XSD schemas that represent the business objects and the corresponding Java data bindings.

The result of running the J2C Bean wizard is a library that contains the Oracle Java data bindings representing business objects and the J2C Bean providing interface and implementation for the functionality of the EIS system. The Java data bindings are used as input and output arguments of the methods of J2C Bean.

Standards Compliance

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

Accessibility

Administration

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

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

You can configure and use product features by utilizing standard text editors and scripted or command line interfaces instead of the graphical interfaces that are provided. When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

J2C Bean wizard

The J2C Bean wizard is the primary component used to create application accessing EIS systems. This wizard is implemented as an Eclipse plug-in that is available through Rational Application Developer for WebSphere Software is fully accessible.

Keyboard navigation

This product uses standard Microsoft Windows® navigation keys.

IBM and accessibility

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

Internet Protocol, Version 6 (IPv6)

IBM WebSphere Application Server, version 6.1.0 and later and its JavaMail component support dual-stack Internet Protocol Version 6.0 (IPv6). For more information about this compatibility in WebSphere Application Server, see IPv6 support in the WebSphere Application Server information center. For more information about IPv6, see http://www.ipv6.org.

Chapter 2. Planning for adapter implementation

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

Before you begin

Before you begin to configure and deploy the module, you should possess a thorough understanding of business integration concepts, Java Database Connectivity (JDBC), the database products in your environment, and the features and capabilities of Rational Application Developer for WebSphere Software and WebSphere Application Server.

To configure and deploy WebSphere Adapter for Oracle E-Business Suite you should understand and have experience with the following concepts, tools, and tasks:

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

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 Oracle database, which might contain sensitive or confidential information. Sometimes these files must 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 situations like this, the adapter lets you specify whether or not you want to prevent confidential user data from displaying in the adapter log and trace files. You can select this option in the J2C Bean wizard or change the HideConfidentialTrace property. When this property is enabled, the adapter replaces the sensitive data with XXX's.

See "Managed connection factory properties" on page 109 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 Oracle 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

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 Application Server.

User authentication

The adapter supports several methods for supplying the user name and password that are needed to connect to the Oracle 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 J2C Bean wizard connects to the Oracle database to extract, or *discover*, information about the objects and services that you can access with the adapter.
- At run time on WebSphere Application Server, when the adapter connects to the Oracle database to process outbound requests and inbound events.

Authentication in the wizard

The J2C Bean 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 Oracle database, although the schema name must be the same in both databases. For example, while developing and integrating an application that uses Adapter for Oracle E-Business Suite, 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 Oracle 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 J2C Bean wizard lets you configure the adapter to get the user information using any of the following methods:

- Adapter properties
- J2C authentication alias

Saving the user name and password in adapter properties is a direct way to provide this information at run time. You provide this user name and password when you use the J2C Bean wizard to configure your module. Although directly specifying the user name and password seems the most straightforward method, it has important limitations. Adapter properties are not encrypted; the password is stored as clear text in fields that are accessible to others on the server. Also, when the password changes, you must update the password in all instances of the adapter that access that Oracle database. This includes the adapters embedded in application EAR files as well as adapters that are separately installed on the server.

Using a J2C authentication data entry, or authentication alias, created with the Java Authentication and Authorization Service (JAAS) feature of Java 2 security is a robust, secure way to deploy applications. An administrator creates the authentication alias that is used by one or more applications that need to access a system. The user name and password 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 will affect the type of deployment option you choose.

The deployment options are described below:

- When you deploy the adapter as an embedded component, the adapter is included within an enterprise application archive (EAR) file and is available only to the application in the EAR file.
- When you deploy the adapter as a stand-alone component, the adapter is represented by a stand-alone resource adapter archive (RAR) file. When it is deployed, it is available to all applications deployed in the server instance.

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

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

If you plan on running multiple versions, and if you care more about potential disruption when you upgrade the adapter, you would be more likely to embed the adapter with the application. Embedding the adapter with the application allows you to associate an adapter version with an application version and administer it as a single module. To deploy the RAR file to the application server, you must obtain and install Adapter for Oracle E-Business Suite (product number 5724-T73). This provides the RAR file that you install following instructions supplied with WebSphere Application Server.

Considerations for embedding an adapter in the application

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

- An embedded adapter has class loader isolation.

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

Considerations for using a stand-alone adapter

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

- Stand-alone adapters have no class loader isolation.

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

  For instance, if you have an adapter that is working with server version X, and you update the version of the client application to version Y, your original application might stop working.
- 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 Application Server, WebSphere Application Server Network Deployment, and WebSphere Extended Deployment support clustered environments. Clusters are groups of servers that are managed together to balance workloads and to provide high availability and scalability. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container to activate the adapter instance. The JCA container provides a runtime environment for adapter instances.

Using WebSphere Extended Deployment, you can optionally enhance the performance of adapter instances in your clustered environment. WebSphere Extended Deployment extends the WebSphere Application Server Network Deployment capabilities by using a dynamic workload manager instead of a static workload manager, which is used by WebSphere Application Server Network Deployment. The dynamic workload manager can optimize the performance of adapter instances in the cluster by dynamically balancing the load of the requests. This means that application server instances can be automatically stopped and started based on the load variations, allowing machines with different capacities and configurations to evenly handle load variations.

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 Oracle database. WebSphere Adapter for Oracle E-Business Suite 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 Oracle E-Business Suite for outbound requests, then you might improve performance by deploying the module to a clustered environment. In a clustered environment, multiple outbound requests can be processed simultaneously, as long as they are not attempting to process the same record.

If multiple outbound requests are attempting to process the same record, such as a Customer address, the workload management capability in WebSphere Application Server Network Deployment distributes the requests among the available adapter instances in the sequence they were received. As a result, these types of outbound requests in a clustered environment are processed in the same manner as those in a single server environment: one adapter instance processes only one outbound request at a time.

Chapter 3. Configuring the module for deployment

To configure the adapter so that it can be deployed on WebSphere Application Server, use Rational Application Developer for WebSphere Software to create a module, which is exported as an EAR file when you deploy the adapter. You then specify the business objects you want to discover and the system on which you want to discover them.

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 store.

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 Oracle database, as follows:
   - ibm_websphere_event_table_create.sql

   The samples are located in the *RAD_installation_dir*/ResourceAdapters/OracleEBS_*version*/Scripts directory, where *RAD_installation_dir* is the installation directory for Rational Application Developer, and *version* identifies the version of the adapter, for example, `6.2.0`.
2. If necessary, set up triggers on user tables so that changes to the user tables can automatically generate events that is stored in the event store. If using triggers is not an option, then you can populate the event store by using custom SQL code or stored procedures that are not invoked through triggers. Instead, these could be invoked through a batch program that runs periodically or that you run manually.

Results

The event store is available for event processing.

Launching the J2C Bean wizard

To begin the process of creating and deploying a module, you start the J2C Bean wizard in Rational Application Developer for WebSphere Software. The wizard creates a connector project, which is used to organize the files associated with the module.

Before you begin

Make sure you have gathered the information you need to establish a connection to the Oracle database. For example, you need the name or IP address of the Oracle 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 J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **File** → **New** → **Other** → **J2C** → **J2C Bean**.
2. Click **Next**.
3. In the Resource Adapter Selection window, expand the Oracle folder and select **IBM WebSphere Adapter for Oracle (IBM :** ***version*****)**, where *version* is the version of the adapter you want to use, for example, 6.2.0.0
4. Click **Next**.
5. In the window, accept the default project name in the **Connector project** field or type a different name.
6. In the **Target server** field, select the type of server where you will deploy the module. The wizard creates the artifacts that are appropriate to that server.
7. Click **Next**. The Connector Settings window is displayed.

Results

A new connector project is created, which contains the adapter RAR file. The project is listed in the Enterprise Explorer view of the Java EE perspective.

What to do next

Continue working in the J2C Bean wizard. The next step is to add database-specific files to the project.

Configuring the connector dependencies

The J2C Bean wizard needs a copy of certain files from the Oracle E-Business Suite to be able to communicate with it. Specify the location of the JAR files that contains the JDBC driver for Oracle and any native system library files that are needed.

Before you begin

You should be running the J2C Bean wizard in Rational Application Developer for WebSphere Software to perform this task.

About this task

In addition to performing this task when you configure the module, you also need to deploy files on WebSphere Application Server.

Procedure

1. Obtain the Oracle 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 following table lists the Oracle JDBC driver files needed for Oracle database software.

Table 11. Oracle JDBC driver files for Oracle database software

Database software	Driver	JDBC driver files	Native System Libraries
Oracle	Thin driver	ojdbc14.jar	None

2. In the Connector Settings window, specify the location of the Oracle JDBC driver-specific files that are required by the adapter.
   a. In **Oracle JDBC driver JAR files**, click **Add** and select the Oracle JDBC driver files.
   b. If you use the Oracle JDBC type 2 driver, click **Add** in **System libraries** to add the native system libraries that are required to access the database server. If you use Oracle JDBC type 4 driver, leave this field empty.
3. Click **Next**. The wizard displays the Adapter Style window.

Results

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

Continue working in the J2C Bean wizard. The next step is to provide the information that the wizard needs to communicate with the Oracle database.

Setting connection properties for the J2C Bean wizard

For the J2C Bean wizard to connect to the database instance to discover database objects, connection properties must be specified.

Before you begin

Before you can configure the connection properties, you must have started the J2C Bean wizard.

About this task

The J2C Bean 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 101.

Procedure

1. In the Adapter Style window, select **Outbound** to pass data from your service import to the adapter or **Inbound** to pass data from the adapter to your service export, and then click **Next**.
2. In the Discovery Configuration window, specify the connection properties for the wizard to use to connect to the Oracle database.
   a. In the list of database software, select your product and version. The **Properties** area displays fields where you specify database-specific connection properties.
   b. In the **JDBC driver type** field, select the type of JDBC driver you want to use.
   c. In the **System ID** field, specify the database name.
   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. If you select a named driver in the **JDBC driver type** field, the wizard provides a default value for **Port number** field. If you select the driver `Other`, the port number is not enabled.

f. If you select a named driver in the **JDBC driver type** field, the wizard provides a default value for the **JDBC driver class name** field and builds the value for the **Database URL** field from other connection fields. If you select the driver `Other`, you must specify the driver class name and database URL (although part of the database URL might be filled in for you).

g. 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. In the **Prefix for business object names** field, type a string to be placed at the beginning of business object names. This prefix field does not require a value, and can be left blank.
5. To set additional advanced properties, click **Advanced**. The related properties are displayed.
6. If you need to set bidirectional properties, perform the following steps:
   a. Expand **Bidi properties** and select the **Bidi transformation** check box.
   b. Set the properties, ordering schema, text direction, symmetric swapping, character shaping, and numeric shaping to control how bidirectional transformation is performed.
7. Optional: If you want to turn AUTOCOMMIT on for the database, select the **Set auto commit on database connection** check box.
8. To change the location of the wizard's log files or the amount of information included in the logs, select the **Specify the level of logging desired** 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 the error messages and the trace information into the standard log and trace files for the server.
9. Click **Next**.

   If the wizard generates the com.ibm.adapter.framework.BaseException exception, the adapter 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 the **Log file output location** field. Make sure that the connection information is correct.

Results

The J2C Bean wizard connects to the database and displays the Object Discovery and Selection window.

What to do next

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

Configuring the module for outbound processing

To configure a module to use the adapter for outbound processing, use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find and select business objects and services from the Oracle database, and to generate business object definitions and related artifacts.

Discovering database objects

After connecting to the database, run a query to search for database objects. Browse the tree of discovered objects to understand the structure of objects in the Oracle 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.

Before you begin

You must understand the data requirements 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, views, synonyms or nicknames, or stored procedures or stored functions you need to access
- How many query business objects you need to create, including parameter values and sample database values for the parameters

About this task

This task starts in the Object Discovery and Selection window of the J2C Bean wizard.

Procedure

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

   Use the Query Filter Parameters 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 business objects you want to create

   The steps for those tasks are given next.
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 **Supported database object types**, and then click **Remove**. If you change your mind, click **Add** to add the object type back. If need to access only specific types of database objects and speed the discovery process, omit the ones you do not need.
4. Select **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. .
5. To create business objects to run user-defined database queries, select **Create a query business object to build user-defined select statements** and then type the number of query business objects you want to create. At this time, you specify only the number of business objects; the wizard prompts for the name and other details about the business objects at a later time.
6. Click **OK** to save your changes to the database query.
7. In the Object Discovery and Selection window, click **Execute Query** to use the query to discover database objects and to create a template for query business objects.

   The **Discovered objects** pane lists the database objects that were discovered.
8. 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, or search a subset of each object type by using filtering.

   To limit the number of objects displayed for a particular object type, highlight a node but do not expand it. Click the **Filtering** icon. 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 object types, such as table or view, which start with that string or match that pattern are displayed under the node.
9. Click **+** (the plus sign) to expand the node for **Query Statements** to display the template for query business objects.

Results

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

What to do next

Continue working in the J2C Bean 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 J2C Bean wizard, and the query object templates you specified, continue using the wizard to select the database objects that you need to access in your module. Then provide configuration information for your new business objects.

About this task

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

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

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

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

     You click on 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 of the objects. The wizard will not close the notebook until you provide all of 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, you cannot use the wizard to add objects to an existing module, so be careful to understand the requirements of the program that uses the business objects before you start the wizard.

Selecting and configuring tables, views, and synonyms or nicknames

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 J2C Bean wizard. You start in the Object Discovery and Selection window and then work in a Configuration Parameters window that is specific to the business object you are configuring.

Procedure

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

   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. Optionally, define a parent-child relationship between business objects.

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

   Configure the parent-child relationship using the area of the Configuration Parameters window. 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 of the table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects.
   b. Specify the cardinality of the relationship:
      - If the table has a single-cardinality relationship with the parent table, select 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 is 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 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 **Preserve *child_table_name* when 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.

c. Click **OK**. The Configuration Parameters 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.
      1) Click **Select**.
      2) In the Select Value window, select the name of the schema you want to work with.
      3) Click **OK**.
   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. If the stored procedure list contains many items, the **Select** button is displayed next to the **Stored procedure name** field. Click **Select** to open the Select window and select the name of the stored procedure or stored function.

      The Configuration Parameters 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.
7. 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 Object Discovery and Selection window.
8. To change the configuration of an object from the **Selected objects** list, select the object name and then click the ✎ (Edit) icon.

In the Object Discovery and Selection window, continue to select and configure other types of business objects. When you are finished, click **Next** to set global properties.

Selecting and configuring stored procedures and stored functions

To select and configure business objects that correspond to stored procedures and stored functions in the database, you filter the database objects, and specify the configuration properties for the database object.

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 J2C Bean wizard. You start in the Object Discovery and Selection window and then work in a Configuration Parameters window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, expand the node for the schema that contains the stored procedure or stored function you want to work with.
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 (Edit or create filter) button, located at the top of the **Discovered objects** pane.
   b. In the Filter Properties window, type a name or pattern in **Object name or pattern**. 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 not 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**. The Stored Procedures node displays all stored procedures whose name matches the catalog filter.
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). The Configuration Properties for 'object' window lists the attributes of the stored procedure business object, which include the names and data type of the parameters of the stored procedure, and information about any result sets that are returned.
4. If the stored procedure returns any result sets, make sure that the value for the **Maximum number of results sets returned from stored procedure** field reflects the maximum number you expect. The wizard creates that number of result set business objects to hold the results.

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

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 procedure does not validate, the adapter will not be able to return the result set at run time.

7. 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 Object Discovery and Selection window.

In the Object Discovery and Selection window, continue to select and configure other types of business objects. When you are finished, click **Next** to set global properties.

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 J2C Bean wizard. You start in the Object Discovery and Selection window and then work in a Configuration Parameters window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, expand the **Query Statements** node. This node contains an object template, named **Select Statement *n***, for each query business object you requested in the Query Filter Parameters window. For example, if you specified a count of two query business objects in that window, the **Discovered objects** list contains two object templates.
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 **Name of the business object**, type a name for the business object. The name can contain spaces and national language characters.

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

   As you type each **?**, the window expands to display an area where you define the WHERE clause for that parameter.
5. In **Where clause parameter *n***, provide information about each parameter in the SELECT statement.
   a. In **Parameter type**, select the data type of the parameter.
   b. In **Sample value**, type a sample value for the parameter.

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

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

Results

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

In the Object Discovery and Selection window, continue to select and configure other types of business objects. When you are finished, click **Next** to set global properties.

Setting global properties for operations

After you select database objects in the J2C Bean wizard, you need to specify properties that apply to all business objects.

Procedure

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

   This window lists all of the operations that the adapter supports for the outbound services for all business objects that you selected on the previous window. Not all operations are supported by each business object. For example, query business objects support only the RetrieveAll operation. Stored procedure 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 change your mind, click **Add** and restore a removed operation.
4. In **Maximum records for RetrieveAll operations**, type the upper limit on the number of records to retrieve for a RetrieveAll operation. The default value is 100. For more information about this property, see “Maximum records for RetrieveAll operation” on page 118.

   Note: If you removed the RetrieveAll operation in step 3, **Maximum records for RetrieveAll operations** will be disabled.
5. In **Business object namespace**, 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.
6. Optionally, in **Folder**, type the relative path to the folder where the generated business objects are to be stored.
7. 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 artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.

- To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
- Otherwise, create a new package:
  a. Click **New**.
  b. In the Java Package window, click **Java project**.
  c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
  d. Accept the default values for the other fields.
  e. Click **Finish**.

3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing "Impl" to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF.`
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. 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.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

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

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

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

Procedure

1. In the Enterprise Explorer in Rational Application Developer for WebSphere Software, expand the Java project, 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, select **Open**. In the editor, go to the Types area and double-click the name of the business object. The editor displays the business object, which has a field for each column.
3. In the editor, select the column you want to set to match another column.
4. In the Properties view, select **Extensions**. If the Properties view is not visible, right-click the column name and click **Show Properties**.
5. Right-click **OracleBusinessObjectTypeMetadata** and then select **New → oracleasi:CopyAttribute**.
6. Select the **CopyAttribute** property.
7. 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`.

8. In the Enterprise Explorer in Rational Application Developer for WebSphere Software, expand the Java project module name, expand src, and then locate the table business object Java file. The business object Java file name is the name of the database schema plus the name of the database table Java file name. For example, J2caCustomer.java.
9. Right-click the business object Java file name and select **Open**.
10. Find the column attribute code line. For example, for `phoneid`, the corresponding line is: `annotation = new LinkedHashMap(); annotation.put("ColumnName", "PHONEID");`
11. Below the code mentioned in the previous step, add the code for new attribueType. For example, if you want to copy from a column in the parent business object, prefix the column name with two periods(..). For example, `annotation.put("CopyAttribute", "..phone");`
12. Save the file.
13. Re-generate the EJB or WEB, and EAR project.

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.

Generating the EJB or JSP project

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

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.

8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the Oracle adapter RAR dependency such as CWYOE_OracleEBS.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the EJB Project window.
     i. Click **Finish** to return to the EJB Creation window.
     j. Click **Next**.

        Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.
     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
     l. Click **Finish**.
9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the Oracle adapter RAR dependency such as CWYOE_OracleEBS.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.

l. Click **Finish**.

10. Export the project as an EAR file for deployment.

Configuring the module for inbound processing

To configure a module to use the adapter for inbound processing, use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find and select business objects and services from the Oracle 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 Oracle database and use filters to display only the database objects you want to see.

Before you begin

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

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

About this task

This task starts in the Object Discovery and Selection window of the J2C Bean wizard.

Procedure

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

   Notice the unavailable option **Create a query business object to build user-defined select statements**. This option is available only for outbound processing.

   Use the Query Filter Parameters window to perform the following tasks:

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

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name 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 **Supported database object types**, and then click **Remove**. If you change your mind, click **Add** to add the object type back. If your database contains object types that you do not need to access, omitting them can speed up the discovery process.
4. Select **Prompt for additional configuration settings when adding business object** 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. .
5. Click **OK** to save your changes to the query.
6. In the Object Discovery and Selection window, click Execute Query to use the query to discover database objects.

   The **Discovered objects** pane lists the objects that were discovered. The tables, views, and synonyms/nicknames are sorted by schema name.
7. In the **Discovered objects** list, click **+** (the plus sign) to expand a schema node and the **Tables**, **Views**, and **Synonyms - Nicknames** nodes underneath it to see the database objects discovered by the wizard.

Results

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

What to do next

Continue working in the J2C Bean 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 J2C Bean wizard, and the query object templates you specified, continue using the wizard to select the database objects that you need to access in your module. Then provide configuration information for your new business objects.

About this task

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

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

1. You select one or more objects in the **Discovered objects** list of the Object Discovery and Selection window.

2. You click the > (Add) button.
3. The wizard opens the Configuration Parameters window.
   - If you selected a single object, a single Configuration Parameters window is displayed.

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

     You click on 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 of the objects. The wizard will not close the notebook until you provide all of 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, you cannot use the wizard to add objects to an existing module, so be careful to understand the requirements of the program that uses the business objects before you start the wizard.

Selecting and configuring tables, views, and synonyms or nicknames

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

Before you begin

To 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 J2C Bean wizard. You start in the Object Discovery and Selection window and then work in a Configuration Parameters window that is specific to the business object you are configuring.

Procedure

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

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. Optionally, define a parent-child relationship between business objects.

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

   Configure the parent-child relationship using the area of the Configuration Parameters window. 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 of the table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects.
   b. Specify the cardinality of the relationship:
      - If the table has a single-cardinality relationship with the parent table, select 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 is 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 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 **Preserve *child_table_name* when 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.

   c. Click **OK**. The Configuration Parameters 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.

      1) Click **Select**.

      2) In the Select Value window, select the name of the schema you want to work with.

      3) Click **OK**.

   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. If the stored procedure list contains many items, the **Select**

button is displayed next to the **Stored procedure name** field. Click **Select** to open the Select window and select the name of the stored procedure or stored function.

The Configuration Parameters 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.

7. 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 Object Discovery and Selection window.
8. To change the configuration of an object from the **Selected objects** list, select the object name and then click the ✎ (Edit) icon.
9. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

What to do next

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

Setting global properties for operations

After you have selected database objects in the J2C Bean wizard, you need to specify properties that apply to all business objects.

Procedure

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

   The specified operations are set for all business objects that are generated.
3. To remove an operation that you do not need, select the operation name and click **Remove**. If you change your mind, click **Add** and restore a removed operation.
4. In **Business object namespace**, accept the default namespace or type the full name of another namespace.

   The namespace is prefixed to the business object name to keep the business object schemas logically separated. For more information about this property, see “Business object namespace (BusinessObjectNameSpace)” on page 132.
5. Optionally, in **Folder**, type the relative path to the folder where the generated business objects are to be stored.
6. 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 artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **EJB Project name** field, select or create a new EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the EJB Project window, type a project name. For example, MyAdapterInboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterInboundEJBEAR
     e. Click **Finish** to return to the EJB Project window.
     f. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the New Java Package window, type a name for the package. For example, myadapterinboundejbpkg.
     c. Click **Finish**.
3. In the **Stateless Session EJB's local business interface name** field, specify the interface name you want to use for your business object. For example, MyAdapterInboundInterface. The interface name is suffixed with “MDB” and it is displayed automatically in the **Message Driven EJB name** field. For example, MyAdapterInboundInterfaceMDB. Similarly, the interface name is suffixed with ″SB″ and it is displayed automatically in the **Stateless Session EJB name** field. For example, MyAdapterInboundInterfaceSB.
4. In the Inbound Connection configuration area, specify the JNDI name, which JNDI data source for an existing activation specification in WAS server or you can create it later. For example, `com/eis/AdapterAS`.
5. Click **Finish**.

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

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

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

Procedure

1. In the Enterprise Explorer in Rational Application Developer for WebSphere Software, expand the Java project, 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, select **Open**. In the editor, go to the Types area and double-click the name of the business object. The editor displays the business object, which has a field for each column.
3. In the editor, select the column you want to set to match another column.
4. In the Properties view, select **Extensions**. If the Properties view is not visible, right-click the column name and click **Show Properties**.
5. Right-click **OracleBusinessObjectTypeMetadata** and then select **New** → **oracleasi:CopyAttribute**.
6. Select the **CopyAttribute** property.
7. 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`.
8. In the Enterprise Explorer in Rational Application Developer for WebSphere Software, expand the project module name, expand ejbModule, and then locate the table business object java file. The business object java file name is the name of the database schema plus the name of the database table java file name. For example, J2caCustomer.java.

9. Right-click the business object Java file name and select **Open**.
10. Find the column attribute code line. For example, for `phoneid`, the corresponding line is: `annotation = new LinkedHashMap(); annotation.put("ColumnName", "PHONEID");`
11. Below the code mentioned in the previous step, add the code for new attribueType. For example, if you want to copy from a column in the parent business object, prefix the column name with two periods(..). For example, `annotation.put("CopyAttribute", "..phone");`
12. Save the file.

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 4. Configuring the application on WebSphere Application Server

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 Application Server.

About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change. The **Configuration** page is displayed.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for Oracle**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **Custom properties**.
8. For each property you want to change, perform the following steps.

   Note: See "Resource adapter properties" on page 106 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.

You can change the number in the **Value** field and add a description of the property.

c. Click **OK**.

9. Click the **Save** link in the **Messages** box at the top of the window.

Results

The resource adapter properties associated with your adapter module are changed.

Setting managed (J2C) connection factory properties for embedded adapters

To set managed connection factory properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Application Server.

About this task

You use managed connection factory properties to configure the target Oracle database instance.

Note: In the administrative console, the properties are referred to as ″J2C connection factory properties.″

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. In the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for Oracle**.
6. In the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C connection factories**.
8. Click the name of the connection factory associated with your adapter module.
9. In the **Additional Properties** list, click **Custom properties**.

   Custom properties are those J2C connection factory properties that are unique to Adapter for Oracle E-Business Suite. 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 109 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter module are changed.

Setting activation specification properties for embedded adapters

To set activation specification properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Application Server.

About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for Oracle**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C activation specifications**.
8. Click the name of the activation specification associated with the adapter module.
9. From the **Additional Properties** list, click **J2C activation specification custom properties**.
10. For each property you want to change, perform the following steps.

    Note: See "Activation specification properties" on page 128 for more information about these properties.

    a. Click the name of the property.

  b. Change the contents of the **Value** field or type a value, if the field is empty.
  c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

Results

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

Changing configuration properties for stand-alone adapters

To set configuration properties after you install a stand-alone adapter, you use the administrative console of the runtime environment. You provide general information about the adapter and then set resource adapter properties (which are used for general adapter operation). If the adapter will be used for outbound operations, you create a connection factory and then set properties for it. If the adapter will be used for inbound operations, you create an activation specification and then set properties for it.

Setting resource adapter properties for stand-alone adapters

To set resource adapter properties for your stand-alone adapter after it has been installed on WebSphere Application Server, 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 Application Server.

About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Oracle**.
4. In the **Additional Properties** list, click **Custom properties**.
5. For each property you want to change, perform the following steps.
   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty. You can change the number in the **Value** field and add a description of the property.
   c. Click **OK**.
6. Click **Save** in the **Messages** box at the top of the page.

Results

The resource adapter properties associated with your adapter are changed.

Setting managed (J2C) connection factory properties for stand-alone adapters

To set managed connection factory properties for your stand-alone adapter after it has been installed on WebSphere Application Server, 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 Application Server.

About this task

You use managed connection factory properties to configure the target Oracle database instance.

Note: In the administrative console, the properties are referred to as ″J2C connection factory properties.″

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Oracle**.
4. In the **Additional Properties** list, click **J2C connection factories**.
5. If you are going to use an existing connection factory, skip ahead to select from the list of existing connection factories.

   Note: If you selected **Use predefined connection properties** when you used the J2C Bean wizard to configure the adapter module, you do not need to create a connection factory.

   If you are creating a connection factory, perform the following steps:

   a. Click **New**.
   b. In the **General Properties** section of the **Configuration** tab, type a name for the connection factory. For example, you could type `AdapterCF`.
   c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterCF`.
   d. Optional: Select an authentication alias from the **Component-managed authentication alias** list.
   e. Click **OK**.
   f. Click **Save** in the **Messages** box at the top of the page.

      The newly created connection factory is displayed.
6. In the list of connection factories, click the one you want to use.
7. In the **Additional Properties** list, click **Custom properties**.

Custom properties are those J2C connection factory properties that are unique to Adapter for Oracle E-Business Suite. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.

8. For each property you want to change, perform the following steps.

   Note: See "Managed connection factory properties" on page 109 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter are set.

Setting activation specification properties for stand-alone adapters

To set activation specification properties for your stand-alone adapter after it has been installed on WebSphere Application Server, 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 Application Server.

About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for Oracle**.
4. In the **Additional Properties** list, click **J2C activation specifications**.
5. If you are going to use an existing activation specification, skip ahead to select from an existing list of activation specifications.

   Note: If you selected **Use predefined connection properties** when you used the J2C Bean wizard to configure the adapter module, you do not need to create an activation specification.

   If you are creating an activation specification, perform the following steps:

   a. Click **New**.

b. In the **General Properties** section of the **Configuration** tab, type a name for the activation specification. For example, you could type `AdapterAS`.

c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterAS`.

d. 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.

6. In the list of activation specifications, click the one you want to use.
7. In the Additional Properties list, click **J2C activation specification custom properties**.
8. For each property you want to set, perform the following steps.

a. Click the name of the property.

b. Change the contents of the **Value** field or type a value, if the field is empty.

c. Click **OK**.

9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the page.

Results

The activation specification properties associated with your adapter are set.

Adding dependency libraries to the deployed resource adapter

The deployed resource adapter running in the WebSphere Application Server requires the same dependency libraries as it does in Rational Application Developer for WebSphere Software to process requests. The method for adding these library files depends on the mode of the resource adapter deployment: standalone or embedded in the EAR file.

Standalone deployment

The dependency libraries may be added to the resource adapter deployed standalone either during initial deployment of the RAR file or by configuring the Resource Adapter properties after deployment.

About this task

To set the values during initial deployment of the RAR file, specify Class path and Native path locations. Class path is used to point to JAR files, and Native path is used to point to native libraries, such as *.dll, *.so.

To set the dependency library path files after the adapter has been installed on WebSphere Application Server, use the administrative console to modify the values for the Resource Adapter.

EAR deployment

For the rare case when the connector needs to be embedded in the EAR file, the dependant libraries are added as shared libraries. Define the appropriate shared library containing external dependencies and associate them with the EAR file.

About this task

There are two methods to do this task:

- Using enhanced EAR editor in Rational Application Developer for WebSphere Software
- Using administrative console of the WebSphere Application Server

Using enhanced EAR editor

You can use the EAR editor in Rational Application Developer for WebSphere Software to add the dependency libraries.

About this task

To create shared libraries using the EAR editor, use the following procedure.

Procedure

1. Open Enhanced EAR editor.
2. Click **Deployment** tab.
3. Expand **Shared Library** Shared Library section.
4. Click **Add** to add new shared library.
5. Specify Shared library parameters and click **OK**.
6. Deploy the EAR to the server.

Using administrative console of the WebSphere Application Server

You can use the administrative console of the WebSphere Application Server to add the dependency libraries.

Before you begin

Make sure dependent files are available on the server machine in the separate folder. If needed, copy dependent files on the server machine.

Procedure

1. Define WebSphere variables to point to appropriate folders.
2. Define the shared library through the server administrative console; you can specify it using WebSphere variables defined in above step 1.
3. Deploy the EAR to the server.
4. Configure the EAR to reference defined shared library.

Chapter 5. Troubleshooting and support

Common troubleshooting techniques and self-help information help you identify and solve problems quickly.

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 `OEBSRA` plus the value of the adapter ID property. For example, if the adapter ID property is set to `001`, the component ID is `OEBSRA001`.

If you run multiple instances of the same adapter, make sure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate 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 log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Oracle E-Business Suite to `001` and `002`. The component IDs for those instances, `OEBSRA001` and `OEBSRA002`, 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. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `OEBSRAInstanc`.

For outbound processing, the adapter ID property is located in both the resource adapter and managed connection factory property groups. If you update the adapter ID property after using the J2C Bean wizard to configure the adapter for outbound processing, be sure to set the resource adapter and managed connection factory properties consistently, to prevent 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 107.

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.

About this task

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 application 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 an application server, including service components and event points, see the documentation for your application server.

You can change the log configuration statically or dynamically. Static configuration takes effect when you start or restart the application server. Dynamic, or runtime, configuration changes apply immediately.

When a log is created, the detail level for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagate the change to their children, as necessary.

To enable logging and set the output properties for a log, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, click **Servers** → **WebSphere application servers**.
2. Click the name of the server that you want to work with.
3. Under **Troubleshooting**, click **Logging and tracing**.
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 Oracle E-Business Suite only, select the **com.ibm.j2ca.oracleebs.*** 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 application 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 application server is written to the SystemOut.log and trace.log files, respectively.

Before you begin

You can change the log and trace file names at any time after the adapter module has been deployed to an application server.

About this task

You can change the log and trace file names statically or dynamically. Static changes take effect when you start or restart the application server. Dynamic or run time changes apply immediately.

Log and trace files are in the *install_root*/profiles/*profile_name*/logs/*server_name* folder.

To set or change the log and trace file names, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, select **Applications > Application Types > WebSphere application servers**.
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 Oracle.
5. In the Configuration tab, under Additional Properties, click **Resource Adapter**.
6. In the Configuration tab, under Additional Properties, click **Custom properties**.
7. In the Custom Properties table, change the file names.
   a. Click either **logFilename** to change the name of the log file or **traceFilename** to change the name of the trace file.
   b. In the Configuration tab, type the new name in the **Value** field. By default, the log file is called SystemOut.log and the trace file is called trace.log.
   c. Click **Apply** or **OK**. Your changes are saved on your local machine.
   d. To save your changes to the master configuration on the server, use one of the following procedures:
      - **Static change**: Stop and restart the server. This method allows you to make changes, but those changes do not take effect until you stop and start the server.
      - **Dynamic change**: Click the **Save** link in the Messages box above the Custom properties table. Click **Save** again when prompted. This method allows you to make changes that take effect right away.

First-failure data capture (FFDC) support

The adapter supports first-failure data capture (FFDC), which provides persistent records of failures and significant software incidents that occur during run time in WebSphere Application Server.

The FFDC feature runs in the background and collects events and errors that occur at run time. The feature provides a means for associating failures to one another, allowing software to link the effects of a failure to their causes, and thereby facilitate the quick location of the root cause of a failure. The data that is captured can be used to identify exception processing that occurred during the adapter run time.

When a problem occurs, the adapter writes exception messages and context data to a log file, which is located in the *install_root*/profiles/*profile*/logs/ffdc directory.

For more information about first-failure data capture (FFDC), see the WebSphere Application Server documentation.

XAResourceNotAvailableException

When the application server log contains repeated reports of the com.ibm.ws.Transaction.XAResourceNotAvailableException exception, remove transaction logs to correct the problem.

Symptom:

When the adapter starts, the following exception is repeatedly logged in the application server log file:

```
com.ibm.ws.Transaction.XAResourceNotAvailableException
```

Problem:

A resource was removed while the application server was committing or rolling back a transaction for that resource. When the adapter starts, it tries to recover the transaction but cannot because the resource was removed.

Solution:

To correct this problem, use the following procedure:

1. Stop the application server.
2. Delete the transaction log file that contains the transaction. Use the information in the exception trace to identify the transaction. This prevents the server from trying to recover those transactions.

   Note: In a test or development environment, you can generally delete all of the transaction logs. In Rational Application Developer for WebSphere Software, delete the files and subdirectories of the transaction log directory, *server_install_directory*\profiles\profile_name\tranlog.

   In a production environment, delete only the transactions that represent events that you do not need to process. One way to do this is to reinstall the adapter, pointing it to the original event database used, and deleting only the transactions you do not need. Another approach is to delete the transactions from either the log1 or log2 file in the following directory:

   server_install_directory\profiles\profile_name\tranlog\node_name\server_name\ transaction\tranlog
3. Start the application server.

Self-help resources

Use the resources of IBM software support to get the most current support information, obtain technical documentation, download support tools and fixes, and avoid problems with WebSphere Adapters. The self-help resources also help you diagnose problems with the adapter and provide information about how to contact IBM software support.

Support Web site

The WebSphere Adapters software support Web site at http://www.ibm.com/software/integration/wbiadapters/support/ provides links to many resources to help you learn about, use, and troubleshoot WebSphere Adapters, including the following types of

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

Recommended fixes

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

Technotes

Technotes provide the most current documentation about the Adapter for Oracle E-Business Suite, 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 Oracle E-Business Suite 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/

Solutions to common problems

Some of the problems you may encounter running WebSphere Adapter for Oracle E-Business Suite with your database are described, along with solutions and workarounds. These problems and solutions are in addition to those documented as technotes on the software support Web site.

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

RecordNotFoundException for RetrieveAll operation in test client

Problem

When performing a RetrieveAll operation in the Rational Application Developer for WebSphere Software test client, the RecordNotFoundException exception is generated when data is expected from the query. The following message is generated: `RecordNotFoundException: Record not found in EIS`.

Cause

This except can occur if the WHERE clause for the SELECT statement does not set all of the attributes of the business object. Leaving the attribute blank, which is the default value, is not the same as explicitly unsetting the value.

Solution

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

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

Problem

You get the following exception when inserting CLOB (character large object) values of 4K and larger into Oracle 9i or 10g databases:

- Oracle 9i: `ResourceAdapt E com.ibm.j2ca.dbadapter.core.runtime.DBOperationHandler. executePreparedCUDStatement CWYDB0301E: 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.dbadapter.core.runtime.DBOperationHandler. executePreparedCUDStatement CWYDB0301E: An operation on the database failed with a SQL exception for the following reason: ORA-01460: unimplemented or unreasonable conversion requested`

Cause

You are using an older driver that does not correctly support CLOBs larger than 4K.

Solution

Use the OracleThin driver from Oracle 10.1.0.2 or a later release.

Some generated business objects have no attributes for Oracle database objects

Problem

For some business objects that are generated from an Oracle database object, the generated business object has no attributes for the table columns.

Cause

Under certain conditions, the Oracle JDBC driver does not return the column information for a database object. The following bugs are currently filed with Oracle for these issues:

- 2281705. DATABASEMETADA.GETCOLUMNS does not return underlying table if there is a synonym

- 2696213. JDBC GETPROCEDURECOLUMNS does not return columns for the synonym of a procedure

Additionally, column information will not be returned if a private synonym that references an object in another schema is used.

Solution

For tables that have a synonym, generate the business object using the synonym for the table.

For synonyms of a procedure, generate the business object using the original procedure that the synonym is based on.

For private synonyms that reference an object in another schema, either use the original table or create a synonym in the current schema.

ResourceException exceptions during outbound processing

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

- SQLException exception

  If the SQLException exception includes the text `UserID or password is invalid`, then the user ID or password specified for the outbound connection is not correct.

  For example:

  ```
  javax.resource.ResourceException: ORA-01017; invalid username/password;
  logon denied.
  ```
- ConnectException exception

  If the text included with the ConnectException exception includes text similar to `is not reachable` or `could not establish the connection`, then the database server might not be operational or there might be a network problem that prevents a connection.

  For example:

  ```
  java.sql.SQLException: Io exception: The Network Adapter could not
  establish the connection
  ```

ResourceException exception during inbound processing

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

UniqueConstraintViolation fault, MultiMatchingRecordsException fault

Class loader violation occurs when starting the J2C Bean wizard

Problem

It is not possible to use the J2C Bean 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 J2C Bean wizard, an error occurs when you attempt to connect to the database from the Data perspective.
- When you establish a connection to the database through the Data perspective, the error occurs when you attempt to connect to database through the J2C Bean wizard.

Cause

The error occurs because the Data perspective and the wizard use their own class loaders. Once the DLL, which is the native library used by the JDBC driver, is loaded in the Data perspective, it cannot be loaded again in the wizard. JVMs have an inherent restriction that only allows one class loader to load native libraries at any given time. So if class loader A loads DLL B, then no other class loader can load DLL B until class loader A is released and garbage is collected. Because you cannot really control garbage collection, it usually means that if you want to load DLL B with another class loader, you need to restart the JVM. This limitation is a known one and is documented for WebSphere Application Server.

Solution

The only solution is to restart Rational Application Developer for WebSphere Software when this error occurs.

Closed connection error occurs when using XA with Oracle 10g

Problem

When the Adapter for Oracle E-Business Suite is used to perform an XA transaction using Oracle 10g, the adapter generates a closed connection exception: javax.resource.ResourceException: Closed Connection.

Cause

This is a known issue with the Oracle 10g database driver. The following bug has been filed with Oracle for this issue: 3488761 Connection closed error from OracleConnection.getConnection() - 10G drivers.

Solution

The bug has been fixed in the Oracle 10g Release 2 driver. As a workaround, you can use the Oracle 9i JDBC Thin drivers to connect to the database for XA transactions.

Error occurs while starting a transaction on Oracle

Problem

When the Adapter for Oracle E-Business Suite is used to perform an XA transaction using Oracle database, the following error is generated: `WTRN0078E: An attempt by the transaction manager to call start on a transactional resource has resulted in an error. The error code was XAER_RMERR.`

Cause

For the Oracle database server to support XA transactions, some commands need to be run.

Solution

Two scripts that are included in the Oracle directory should be run. This activity will likely need to be performed by your Oracle database administrator, because you must be logged into Oracle as SYSOPER or SYSDBA in order to have the necessary permissions for these scripts to work. The scripts are:

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

The initxa.sql script configures the database for XA. Once it runs successfully, your database is configured for XA. The script might run successfully the first time you try. Unfortunately, it probably will not run successfully because some of the database's memory spaces are too small.

To fix this, run the initjvm.sql script. It will probably fail too, but in doing so, it will indicate which parameters need to be adjusted. The parameters are stored in this file:

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

Table 12 shows two parameters that typically need to be increased. Your particular database configuration might require adjustment of different parameters.

Table 12. Typical parameter sizes

Parameter Name	Minimum Value
java_pool_size	12000000
shared_pool_size	24000000

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

Should you receive unexpected results from a query, turn on tracing 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 not fill in your input business object correctly.

Chapter 6. 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 should process the business object as well as the operation to be performed on the business object. The name of the business object is generated by the J2C Bean wizard in accordance with the naming convention for the adapter.

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.

A business object is simply a container for the data specified in the attributes. Each attribute has a name, type, cardinality, and several other properties. The J2C Bean wizard sets the attribute name to the name of the column. The adapter adds the attribute cardinality, type, and application-specific information. 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 13 lists the properties of a business object attribute and describes their interpretation and settings.

Table 13. 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 or multiple (an unbounded integer) cardinality, respectively. In both single- and multiple-cardinality relationships, the relationship between the parent and child business objects is described by the application-specific information of the key attribute in the business object storing the relationship.
Foreign Key	When arrays of child business objects whose cardinality is *n* are retrieved, foreign keys are used in the WHERE clause of SELECT statements. The RetrieveAll operation overrides the use of keys and foreign keys. **Note:** The adapter does not support specifying an attribute that represents a child business object as a foreign key.
Name	This property represents the unique name of the attribute, if it is a simple attribute, or the name of the business object, if it is a child business object.

Table 13. Attribute properties (continued)

Properties	Interpretation and settings
MinOccurs MaxOccurs	If the column is not a primary key and is not null able, 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 Oracle type of a database object are described in Table 14. 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 Oracle metadata, maps to the business object attribute types as listed in Table 14. Only the Oracle 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`.

Note: On generating the business object for stored procedures or tables, if the Oracle metadata does not map to the same business object attribute type, update the attribute data type manually in the XSD file for the business object.

Table 14. Oracle metadata column type and business object attribute types

Oracle metadata column type	Business object attribute type
CHAR LONG VARCHAR2	String
NUMBER	String

Table 14. Oracle metadata column type and business object attribute types (continued)

Oracle metadata column type	Business object attribute type
TIMESTAMP DATE	String
FLOAT	Double
BLOB	hexBinary
CLOB	String
NCHAR NVARCHAR2	String
RAW LONG RAW	hexBinary

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 15.

Table 15. Application-specific information for simple attributes

Parameter	Type	Description	Default value
BLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the BLOB data type. While displaying BLOB data, the adapter displays the number of bytes as a hexadecimal value. The attribute type is hexBinary. If `True`, the column data type is BLOB.	None
ByteArray	Boolean	Specifies whether the column is a binary data type. If `True`, the adapter reads and writes binary data to the database and sends that data as a string to the application server. The adapter sets binary data on the business object. The attribute type is hexBinary.	`False`
ChildBOType	String	If the attribute is a complex data type, use this application-specific information to specify the actual type: • Struct • Array • ResultSet	None
ChildBOTypeName	String	When the value of the ChildBOType application-specific information is either Struct or Array, this parameter value represents the name of the user-defined type. This value is case-sensitive.	

Table 15. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
CLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the CLOB data type. This value applies only to attributes of type String. If `True`, the column data type is CLOB. The CLOB attribute has a String Type whose length is used to define the length of the CLOB.	None
ColumnName	String	The name of the database column corresponding to this attribute. This is the only required parameter.	None
CopyAttribute	String	A user-specified value that refers to another attribute name from within the same business object or parent business object. If the value set in the application-specific information refers to the name of another attribute within the same business object, then the adapter uses the value of the other attribute to set the value of this attribute (on which application-specific information is defined) before it adds the business object to the database during a Create operation. For example, if you want the `contact` column of a new row in the table to contain the same value as the `email` column, set the CopyAttribute parameter of the `contact` attribute to `email`. The value cannot reference an attribute in a child business object, but it can reference an attribute in the parent business object by preceding the name with two periods. For example, you can reference the `ccode` attribute in a parent business object as `..ccode`. If you do not include this parameter in the application-specific information, the adapter uses the value of the current attribute without copying the value from another attribute.	None
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 adapter uses the Timestamp value present in the database. To learn more about the Timestamp method, go to the Sun web site at http://java.sun.com/j2se/1.5.0/docs/api/ and search for *Timestamp*.	None

Table 15. 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 for the business object. Open the business object by using an XML or text editor to edit the XSD file. Two changes must be made, as follows:
		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>` tag, as in this example:
		`<simpletype>`
		`<restriction base="string">`
		`<maxLength value="10"/>`
		`</restriction>`
		`</simpleType>`
		Save the object definition, and ensure that no validation errors occur in the XSD file after it has been updated.
		Following this table, see the Example of FixedChar parameter in the business object XSD file.
			`false`

ForeignKey	String	The value of this property depends on whether the parent/child relationship is stored in the parent business object or in the child. If the relationship is stored in the parent, the value includes both the type of the child business object and the name of the attribute in the child to be used as the foreign key *(Child_BO_name/Child_Property_Name)*. If the relationship is stored in the child, set the value to include only the name of the attribute in the parent to be used as the foreign key. If an attribute is not a foreign key, do not include this parameter in the application-specific information.	None
OrderBy	String	If a value is specified and the attribute is in a child business object, the adapter uses the value of the attribute in the ORDER BY clause of retrieval queries. The adapter can retrieve child business objects in either ascending order (ASC) or descending order (DESC). If you do not include this parameter in the application-specific information, the adapter does not specify the retrieval order.	None
PrimaryKey	Boolean	If the column associated with this attribute is a primary key in the corresponding table in the database, PrimaryKey is to `True`,	None

Table 15. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
SPParameterType	String	Specifies the type of stored procedure Possible values are: • IP (input only) • OP (output only) • IO (input and output) • RS (result set)	None
UniqueIdentifier (UID)	String	The adapter uses this parameter to generate the unique ID for the business object. It supports the generation of sequences and identity columns. The format of this parameter is as follows: UID=AUTO|*Sequence_Name* For a sequence, set the UID attribute to the name of the sequence. For an identity column, set the UID attribute to AUTO. 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

```
<element name="pkey" nillable="true"
                        minOccurs="0" maxOccurs="1">
                        <annotation xml:space="preserve">
                  <appinfo

       source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">

                        <oracleasi:OracleAttributeTypeMetadata
       xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/
oracle/metadata">

       <oracleasi:ColumnName>PKEY</oracleasi:ColumnName>

       <oracleasi:PrimaryKey>true</oracleasi:PrimaryKey>

       <oracleasi:FixedChar>true</oracleasi:FixedChar>
                         </oracleasi:OracleAttributeTypeMetadata>
                   </appinfo>
            </annotation>
            <simpleType>
            <restriction base="string">
            <maxLength value="10"/>
            </restriction>
         </simpleType>
                  </element>
                  <element name="ccode" type="string" nillable="true"
                        minOccurs="0" maxOccurs="1">
                        <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
```

```
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:ColumnName>CCODE</oracleasi:ColumnName>
<oracleasi:PrimaryKey>false</oracleasi:PrimaryKey>
<oracleasi:ForeignKey>custinfoobj/ccode</oracleasi:ForeignKey>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
                  </element>
                  <element name="fname" type="string" nillable="true"
                        minOccurs="0" maxOccurs="1">
                        <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:ColumnName>FNAME</oracleasi:ColumnName>
<oracleasi:PrimaryKey>false</oracleasi:PrimaryKey>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
                  </element>
                  <element name="lname" type="string" nillable="true"
                        minOccurs="0" maxOccurs="1">
                        <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:ColumnName>LNAME</oracleasi:ColumnName>
<oracleasi:PrimaryKey>false</oracleasi:PrimaryKey>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
                  </element>
                  <element name="custinfoobj"
                        type="rtassercustinfo:RtasserCustinfo"
nillable="true" minOccurs="0"
                        maxOccurs="1">
                        <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:Ownership>true</oracleasi:Ownership>
<oracleasi:KeepRelationship>false</oracleasi:KeepRelationship>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
                  </element>
```

Example of FixedChar parameter in the business object XSD file

```
<element name="pkey" nillable="true"
                        minOccurs="0" maxOccurs="1">
                        <annotation xml:space="preserve">
                  <appinfo

       source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
                         <oracleasi:OracleAttributeTypeMetadata

       xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/
oracle/metadata">
```

```
        <oracleasi:ColumnName>PKEY</oracleasi:ColumnName>

        <oracleasi:PrimaryKey>true</oracleasi:PrimaryKey>

        <oracleasi:FixedChar>true</oracleasi:FixedChar>
                          </oracleasi:OracleAttributeTypeMetadata>
                    </appinfo>
              </annotation>
              <simpleType>
              <restriction base="string">
              <maxLength value="10"/>
              </restriction>
           </simpleType>
</element>
```

Application-specific information for attributes of type child business object

Two application-specific information parameters are used for attributes that refer to child business objects (complex, as opposed to simple, attributes). When you set this application-specific information, specify the parameters listed in Table 16.

Table 16. Application-specific information for attributes of type child business object

Parameter	**Type**	**Description**	**Default value**
KeepRelationship	Boolean	If True, this parameter prevents the deletion of a child business object during an Update operation.	None
Ownership	Boolean	This parameter specifies that a child business object is owned by the parent. If True, then Create, Update, and Delete operations on the child business object are allowed. If False, then no updates can be applied to the child business object. When its parent is created, the existence of the child is validated to ensure that relationship integrity is maintained in the database.	None

Example of ownership in the business object XSD file

```
<element name="addressobj"
                        type="rtasseraddress:RtasserAddress"
nillable="true"
                        minOccurs="0"
                        maxOccurs="unbounded">
                        <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:Ownership>true</oracleasi:Ownership>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
</element>

<element name="custinfoobj"
                         type="rtassercustinfo:RtasserCustinfo"
nillable="true"                                  minOccurs="0"
                         maxOccurs="1">
                         <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
```

```
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:Ownership>false</oracleasi:Ownership>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
```

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

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

```
<element name="addressobj"
                        type="rtasseraddress:RtasserAddress"
nillable="true"
                        minOccurs="0"
                        maxOccurs="unbounded">
                       <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:Ownership>true</oracleasi:Ownership>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
</element>

<element name="custinfoobj"
                         type="rtassercustinfo:RtasserCustinfo"
nillable="true"                                 minOccurs="0"
                         maxOccurs="1">
                         <annotation xml:space="preserve">
<appinfo
source="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/metadata">
<oracleasi:OracleAttributeTypeMetadata
xmlns:oracleasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle/
metadata">
<oracleasi:Ownership>false</oracleasi:Ownership>
</oracleasi:OracleAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
```

Application-specific information for operations

The adapter uses application-specific information at the operation level to perform operations, such as to retrieve and update information in the database. The adapter retrieves and updates database tables using SQL queries, stored procedures, or stored functions, as specified in the business objects.

If you choose to add stored procedures or stored functions to the business objects, set the operation application-specific information (ASI) as specified in Table 17 on page 96.

Table 17. 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 the business object attribute. This value indicates that procedure returns the value that is to be assigned to that particular attribute in the business object at run time. If the attribute is another child business object, the adapter returns an error.
StoredProcedure	Yes	Set to the stored procedure name.
StoredProcedureType	Yes	You choose from a list of types. For information about valid stored procedure types, see “Stored procedure type” on page 25.
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 J2C Bean 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 18 on page 97.

Table 18. 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 J2C Bean 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 J2C Bean 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 19.

Table 19. 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 StatusColumnName 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 J2C Bean 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 20 on page 98. 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 J2C Bean 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 20. Business object application-specific information (ASI) for business objects based on stored procedures

Application-specific information	Type	Description
SPName	String	The name of the stored procedure or stored function
ResultSet	Boolean	Indicates whether the stored procedure or stored function will return a result set. If `true`, the stored procedure returns one or more result sets. If `false`, the stored procedure or stored function does not return a result set.
MaxNumberOfRetRS	String	The maximum number of returned result sets that are handled by the adapter runtime
ReturnValue	String	Set to the name of the corresponding business object attribute if the stored procedure has a return value. If the returned value is of simple data type, the attribute is also of simple data type. If the returned value is a result set, this attribute points to a child business object.

Application-specific information for query business objects

For query business objects, there is one business object-level application-specific information, as shown in Table 21.

Table 21. 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 J2C Bean wizard.

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.

Naming conventions

When the J2C Bean 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 J2C Bean 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 `Oracle` or *`%AppName%`* as a prefix to help distinguish between two types of business objects: application-specific and generic. The remainder of the name can describe the table or stored procedure that the business object represents. For example, if the business object definition is generated for the Employee Table in a database application, such as Human Resources (HR), the respective business object name will be `HREmployee`.

For business objects that do not correspond to database objects, such as business objects for database queries, if you give the business object the same name as a table or stored procedure business object, a different number is appended at the end of each name to differentiate them and avoid overwriting.

Globalized characters are supported in any business object name.

You can rename business objects by using the refactoring functionality in Rational Application Developer for WebSphere Software. For more details, refer to the Rational Application Developer for WebSphere Software documentation.

The following table describes the naming conventions that the wizard uses for the business object.

Table 22. Business object naming conventions

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 J2C Bean wizard generates the name of the business object in the form of *Prefix* + *SchemaName* + *ObjectName*, where: • *Prefix* is the value as specified in the J2C bean connection property named Prefix. A prefix is not required, and if not specified, no prefix will be added to the business object name. • *SchemaName* is the name of the schema to which the object belongs. • *ObjectName* is the name of the table, view, stored procedure, stored function, or synonym/nickname. A number is appended 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.

Table 22. Business object naming conventions (continued)

Element	Naming convention
Query business objects	For query business objects, the J2C Bean wizard generates the name of the business object in the form of *Prefix* + *QueryBOName*, where: • *Prefix* is the prefix you specify in the wizard. A prefix is not required, and if not specified, no prefix will be added to the business object name. • *QueryBOName* is the value you specified when you configured the business object in the wizard. A number is appended if necessary to differentiate the business object from another business object with the same name.

Outbound configuration properties

WebSphere Adapter for Oracle E-Business Suite has several categories of outbound connection configuration properties, which you set with the J2C Bean wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Application Server using Rational Application Developer for WebSphere Software or the administrative console, but connection properties for the J2C Bean wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for Oracle E-Business Suite are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	Explanation
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the J2C Bean wizard provides a default value for required properties. Removing a default value from a required field on the J2C Bean wizard *will not change that default value*. When a required field contains no value at all, the J2C Bean wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	The predefined value that is set by the J2C Bean wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table 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.

Row	Explanation
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For 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), Codepage number is set to `8000`″.
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

J2C Bean connection properties are used to establish a connection between the J2C Bean wizard, a tool that is used to create business objects, and the database.

The J2C Bean properties specify such things as connection configuration, bidirectional transformation properties, and logging options for the wizard. After a connection is established, the wizard can discover in the database the metadata it needs to create business objects. Some of the properties that you provide for the wizard to use to discover objects in the database are used as the initial value for the runtime properties that you specify later in the wizard. These include resource adapter, managed connection factory, and activation specification properties.

The connection properties for the J2C Bean wizard are described in the following table. A more detailed 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 100.

Table 23. Connection properties for the J2C Bean 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 software"	The name and version of the database management software that the adapter will access
Database URL	The database URL that is used to connect to the database
"Host name" on page 103	The host name or IP address of the database server
JDBC drive class name	The name of the JDBC driver class
"JDBC driver type" on page 104	The type of JDBC driver to use
Password	Password for the corresponding user name
"Port number" on page 105	The port number for connecting to the database instance
"Prefix for business object names" on page 105	A prefix to be added to the name of the business object
System ID	The system ID (SID) name of the database
User name	The database user name for connecting to the database

The J2C Bean 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 24. Additional JDBC driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties 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: `loginTimeout:20; ConnectionRetryCount:5; ConnectionRetryDelay:5`
Globalized	Yes
Bidi supported	No

Database software

This property specifies the software that manages the database that the adapter will access.

Table 25. Database software details

Row	Explanation
Required	Yes
Possible values	Oracle database software by name and version number.
Default	No default value
Property type	String
Usage	The J2C Bean 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 `Oracle 10`, the JDBC driver class field in the wizard displays only the JDBC drivers supported by that version of Oracle database.
Globalized	Yes
Bidi supported	Yes

Database URL

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

Table 26. Database URL details

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 Oracle database servers. `jdbc:oracle:thin:@9.26.248.148:1521:dev`
Globalized	Yes
Bidi supported	Yes

Host name

This property specifies the host name or IP address of the database server.

Table 27. 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 28. 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 29. 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	Although you are essentially selecting either a type 2 or type 4 (universal) driver, each database system has its own name for the driver. The wizard displays a list of drivers known for each database system, Select `Other` if your driver is not listed. The information in this field must agree with the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 30. 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

Table 30. Password details (continued)

Row	**Explanation**
Bidi supported	Yes

Port number

This property specifies the port number for the database instance.

Table 31. Port number details

Required	Yes
Default	The default value is database-specific and is initialized by the wizard if you select a named driver in the **JDBC driver type** field. If you select `Other` for the driver type, a default value is not provided.
Property type	String
Usage	This is the port number for connecting to the database instance. This property is not enabled if you select `Other` in 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 32. Prefix details

Required	No
Default	No default value
Property type	String
Usage	Use a prefix to help distinguish between the types of business objects.
Example	You might specify a prefix of `Oracle` for generic business objects and `%AppName%` for an application-specific business object.
Globalized	Yes
Bidi supported	No

System ID

This property specifies the system ID (SID) name of the database.

Table 33. System ID details

Required	Yes
Default	The default value is database-specific.
Property type	String
Usage	The system ID (SID) is the name of the database.
Globalized	Yes
Bidi supported	Yes

User name (UserName)

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

Table 34. User name details

Required	Yes
Default	No default value
Property type	String
Usage	The user name is the name entered to connect to the database for the purpose of 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 J2C Bean wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these 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 100.

Table 35. Resource adapter properties for the Adapter for Oracle E-Business Suite

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
Disguise user data as ″XXX″ in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
(Not available)	enableHASupport	Do not change this property.
(Not available)	LogFileMaxSize	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)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

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

Table 36. Adapter ID details

Required	Yes
Default	`001`
Property type	String
Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `OEBSRA`, 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 `OEBSRA001`. If you run multiple instances of the same adapter, make sure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate 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 log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Oracle E-Business Suite to `001` and `002`. The component IDs for those instances, `OEBSRA001` and `OEBSRA002`, 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. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `OEBSRAInstanc`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

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

Table 37. Database vendor details

Required	Yes
Possible values	`Oracle`
Default	`ORACLE`
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. Specify `Oracle` as the value that corresponds to your database vendor.
Globalized	No

Table 37. Database vendor details (continued)

Bidi supported	No

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

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 38. 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, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

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

Query timeout (QueryTimeOut)

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

Table 39. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 40. 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 populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

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

Table 41. 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

Managed connection factory properties

Managed connection factory properties are used by the adapter at run time to create an outbound connection instance with the Oracle database.

You set managed connection factory properties using the J2C Bean wizard during adapter configuration. You can change them using the Rational Application Developer for WebSphere Software editor or, after deployment, with the WebSphere Application Server administrative console.

The following table lists and describes the managed connection factory properties. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 100.

Note: The J2C Bean wizard refers to these properties as managed connection factory properties, while the administrative console refers to them as J2C connection factory properties.

Table 42. Managed connection factory properties for Adapter for Oracle E-Business Suite

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	DriverConnectionProperties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
Auto commit	AutoCommit	The AutoCommit value to use on the connection
Data source JNDI name	DataSourceJNDIName	The name of the JNDI data source to use to establish a connection to the database
Database URL	DatabaseURL	The database URL that is used to connect to the database
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
Disguise user data as ″XXX″ in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X’s instead of user data in log and trace files
JDBC driver class	JDBCDriverClass	The class name of the JDBC driver that is used to connect to the database
Password	Password	Password for the corresponding user name
Query timeout	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
SQL query to connection	PingQuery	The SQL query used to test the reliability of the connection to the database
User name	UserName	The database user name
XA data source name	XADataSourceName	The name of the XA data source to use to establish an connection to the database for XA (distributed) transactions

Adapter ID (AdapterID)

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

Table 43. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 43. Adapter ID details (continued)

Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `OEBSRA`, 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 `OEBSRA001`. If you run multiple instances of the same adapter, make sure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate 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 log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Oracle E-Business Suite to `001` and `002`. The component IDs for those instances, `OEBSRA001` and `OEBSRA002`, 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. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `OEBSRAInstanc`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes
Bidi supported	No

Additional JDBC driver connection properties (DriverConnectionProperties)

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

Table 44. Additional JDBC driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties 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: `loginTimeout:20; ConnectionRetryCount:5; ConnectionRetryDelay:5`
Globalized	Yes
Bidi supported	No

Auto commit (AutoCommit)

This property specifies whether AutoCommit is set for the connection.

Table 45. Auto commit details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	This property is ignored for XA (distributed) transactions.
Globalized	No
Bidi supported	No

Data source JNDI name (DataSourceJNDIName)

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

Table 46. Data source JNDI name details

Required	No
Default	No default value
Property type	String

Table 46. Data source JNDI name details (continued)

Usage	Use this property to specify the JNDI name of a data source in WebSphere Application Server 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. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName property, if set, to establish the connection. The DataSourceJNDIName property represents an XA or connection pool data source. If you define a JNDI data source on the server that supports XA transactions and then specify that data source when you configure the adapter, the adapter participates in XA transactions. Optionally, you can specify the XADataSourceName property for the adapter to participate in XA transactions. 3. If the DataSourceJNDIName and XADataSourceName properties are not set, the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. Do not confuse the data source JNDI name property with the JNDI name of a managed connection factory or activation specification on the server. The following list highlights important differences between the types of JNDI names: • Data source JNDI name – Specifies a connection to a database – 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 47. Database URL details

Required	Yes, unless any of the following properties or sets or properties are set: • The DataSourceJNDIName • The XADataSourceName property
Default	No default value

Table 47. Database URL details (continued)

Property type	String
Usage	In the J2C Bean wizard, compose the database URL by filling in database-specific fields. For example, the database URL for an Oracle database is composed of the system ID (SID), server host name, and database port number. In the administrative console, type the complete database URL value. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName property, if set, to establish the connection. The DataSourceJNDIName property represents an XA or connection pool data source. If you define a JNDI data source on the server that supports XA transactions and then specify that data source when you configure the adapter, the adapter participates in XA transactions. Optionally, you can specify the XADataSourceName property for the adapter to participate in XA transactions. 3. If the DataSourceJNDIName and XADataSourceName properties are not set, the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets (`[]`).
Examples	The following example illustrates typical database URL values for a common database:`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 48. Database vendor details

Required	Yes
Possible values	`Oracle`
Default	`ORACLE`
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. Specify `Oracle` as the value that corresponds to your database vendor.
Globalized	No
Bidi supported	No

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

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 49. 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, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

JDBC driver class (JDBCDriverClass)

This property specifies the class name of the JDBC driver that is used to connect to the database.

Table 50. JDBC driver class details

Required	Yes, if the DataSourceJNDIName property is not set
Possible values	The value specified when you set the connection properties to connect to the database instance.
Default	No default value
Property type	String
Usage	In the J2C Bean wizard, the JDBC driver class is specified for you if you select a common database software and driver combination, such as type 4 drivers for recent versions of Oracle. For type 2 drivers , you must type the database class name. If you select a type 2 driver or a generic driver, you must type the JDBC driver class name. In the administrative console, type the database-specific name of the driver. If the DataSourceJNDIName property is set, this property is ignored.
Examples	Values for the JDBC driver class display in both the J2C Bean wizard and the administrative console. The following examples depict the JDBC driver class properties in both the J2C Bean wizard and the administrative console.**In the J2C Bean wizard:** • To connect to an Oracle 10 database using the type 4 driver, select `Oracle Thin Driver`. **In the administrative console:** **Oracle Thin JDBC driver** `oracle.jdbc.driver.OracleDriver`
Globalized	No
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 51. Password details

Required	No. If you set the Authentication alias or the DataSourceJNDIName, the password is not mandatory. However, if you set the Authentication alias, the DataSourceJNDIName, and Password field, the value specified for the Password takes precedence.
Default	No default value
Property type	String
Usage	If set, this property overrides the password specified on a data source on the server using the Authentication alias or DataSourceJNDIName property.
Globalized	Yes
Bidi supported	Yes

Query timeout (QueryTimeOut)

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

Table 52. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 53. Return business object even when the stored procedure result set is empty details

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

Table 53. Return business object even when the stored procedure result set is empty details (continued)

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 populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

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

Table 54. 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 55. User name details

Required	No. If you set the Authentication alias or the DataSourceJNDIName, the User name property is not mandatory. However, if you set the Authentication alias, the DataSourceJNDIName, and User name, the value specified for the User name takes precedence.
Default	No default value
Property type	String
Usage	Setting this property overrides the user name specified on a data source on the server using the DataSourceJNDIName property or the Authentication alias.
Globalized	Yes
Bidi supported	Yes

XA data source name (XADataSourceName)

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

Table 56. XA data source name details

Required	No
Default	No default value
Property type	String
Usage	To make an XA connection to an Oracle database, this property is used. If the DataSourceJNDIName property is specified, this property is ignored.
Examples	A typical value for an Oracle database: `oracle.jdbc.xa.client.OracleXADataSource`
Globalized	No
Bidi supported	No

Interaction specification properties

Interaction specification, or InteractionSpec, properties control the interaction for an operation. The J2C Bean 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 editor in Rational Application Developer for WebSphere Software. The properties reside in the method binding of the import.

Table 57 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 100.

Table 57. Interaction specification property for the Adapter for Oracle E-Business Suite

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 58. Maximum records for RetrieveAll operation details

Required	Yes
Default	100
Usage	If the number of matches in the database exceeds the value of this property, the adapter throws the MatchesExceededLimitException exception and MatchesExceededLimitFault fault. If a RetrieveAll operation does not return all of the records, increase the value. If you experience out-of-memory issues, reduce the value.

Table 58. Maximum records for RetrieveAll operation details (continued)

Property type	Integer
Globalized	No
Bidi supported	No

Inbound configuration properties

WebSphere Adapter for Oracle E-Business Suite has several categories of inbound connection configuration properties, which you set with the J2C Bean wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using Rational Application Developer for WebSphere Software or the administrative console, but connection properties for the J2C Bean wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for Oracle E-Business Suite are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	Explanation
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the J2C Bean wizard provides a default value for required properties. Removing a default value from a required field on the J2C Bean wizard *will not change that default value*. When a required field contains no value at all, the J2C Bean wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	The predefined value that is set by the J2C Bean wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer

Row	Explanation
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), Codepage number is set to 8000".
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

J2C Bean connection properties are used to establish a connection between the J2C Bean wizard, a tool that is used to create business objects, and the database.

The J2C Bean properties specify such things as connection configuration, bidirectional transformation properties, and logging options for the wizard. After a connection is established, the wizard can discover in the database the metadata it needs to create business objects. Some of the properties that you provide for the wizard to use to discover objects in the database are used as the initial value for the runtime properties that you specify later in the wizard. These include resource adapter, managed connection factory, and activation specification properties.

The connection properties for the J2C Bean wizard are described in the following table. A more detailed 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 100.

Table 59. Connection properties for the J2C Bean 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 software" on page 121	The name and version of the database management software that the adapter will access
Database URL	The database URL that is used to connect to the database

Table 59. Connection properties for the J2C Bean wizard (continued)

Property name in the wizard	Description
"Host name" on page 122	The host name or IP address of the database server
JDBC drive class name	The name of the JDBC driver class
"JDBC driver type" on page 123	The type of JDBC driver to use
Password	Password for the corresponding user name
"Port number" on page 124	The port number for connecting to the database instance
"Prefix for business object names" on page 124	A prefix to be added to the name of the business object
System ID	The system ID (SID) name of the database
User name	The database user name for connecting to the database

The J2C Bean 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 60. Additional JDBC driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties 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: `loginTimeout:20; ConnectionRetryCount:5; ConnectionRetryDelay:5`
Globalized	Yes
Bidi supported	No

Database software

This property specifies the software that manages the database that the adapter will access.

Table 61. Database software details

Row	Explanation
Required	Yes
Possible values	Oracle database software by name and version number.
Default	No default value
Property type	String

Table 61. Database software details (continued)

Row	Explanation
Usage	The J2C Bean 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 `Oracle 10`, the JDBC driver class field in the wizard displays only the JDBC drivers supported by that version of Oracle database.
Globalized	Yes
Bidi supported	Yes

Database URL

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

Table 62. Database URL details

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 Oracle database servers. `jdbc:oracle:thin:@9.26.248.148:1521:dev`
Globalized	Yes
Bidi supported	Yes

Host name

This property specifies the host name or IP address of the database server.

Table 63. 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 64. JDBC driver class name details

Row	Explanation
Required	Yes

Table 64. JDBC driver class name details (continued)

Row	Explanation
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 65. 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	Although you are essentially selecting either a type 2 or type 4 (universal) driver, each database system has its own name for the driver. The wizard displays a list of drivers known for each database system, Select `Other` if your driver is not listed. The information in this field must agree with the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 66. 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 67. Port number details

Required	Yes
Default	The default value is database-specific and is initialized by the wizard if you select a named driver in the **JDBC driver type** field. If you select `Other` for the driver type, a default value is not provided.
Property type	String
Usage	This is the port number for connecting to the database instance. This property is not enabled if you select `Other` in 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 68. Prefix details

Required	No
Default	No default value
Property type	String
Usage	Use a prefix to help distinguish between the types of business objects.
Example	You might specify a prefix of `Oracle` for generic business objects and `%AppName%` for an application-specific business object.
Globalized	Yes
Bidi supported	No

System ID

This property specifies the system ID (SID) name of the database.

Table 69. System ID details

Required	Yes
Default	The default value is database-specific.
Property type	String
Usage	The system ID (SID) is the name of the database.
Globalized	Yes
Bidi supported	Yes

User name (UserName)

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

Table 70. User name details

Required	Yes
Default	No default value

Table 70. User name details (continued)

Property type	String
Usage	The user name is the name entered to connect to the database for the purpose of 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 J2C Bean wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these 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 100.

Table 71. Resource adapter properties for the Adapter for Oracle E-Business Suite

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
Disguise user data as ″XXX″ in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
(Not available)	enableHASupport	Do not change this property.
(Not available)	LogFileMaxSize	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)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

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

Table 72. Adapter ID details

Required	Yes
Default	001
Property type	String
Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, OEBSRA, 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 OEBSRA001. If you run multiple instances of the same adapter, make sure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate 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 log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for Oracle E-Business Suite to 001 and 002. The component IDs for those instances, OEBSRA001 and OEBSRA002, 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. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to OEBSRAInstanc. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

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

Table 73. Database vendor details

Required	Yes
Possible values	Oracle
Default	ORACLE
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. Specify Oracle as the value that corresponds to your database vendor.
Globalized	No
Bidi supported	No

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

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 74. 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, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

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

Query timeout (QueryTimeOut)

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

Table 75. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 76. 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 populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

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

Table 77. 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

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 J2C Bean wizard during adapter configuration and can change them using the Rational Application Developer for WebSphere Software editor or, after deployment, the WebSphere Application Server 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 100.

Table 78. Activation specification properties for Adapter for Oracle E-Business Suite

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	DriverConnectionProperties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
Business object namespace	BusinessObjectNameSpace	The namespace for the business object definitions
Custom delete query	CustomDeleteQuery	The name of the query, stored procedure, or stored function that is run after each event is processed to delete records that can be deleted after the event is delivered
Custom event query	CustomEventQuery	The name of the query, stored procedure, or stored function that performs the polling for events
Custom update query	CustomUpdateQuery	The name of the query, stored procedure, or stored function that is run after each event is processed to prevent the event from being picked up for processing in a subsequent event cycle
Data source JNDI name	DataSourceJNDIName	The name of the JNDI data source to use to establish a connection to the database
Database URL	DatabaseURL	The database URL that is used to connect to the database
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
Do not process events that have a timestamp in the future	FilterFutureEvents	Specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time
Ensure once-only event delivery	AssuredOnceDelivery	Specifies whether the adapter provides assured once delivery of events
Event order by	EventOrderBy	The order in which events are retrieved and processed
Event query type	EventQueryType	Determines whether to use the standard event store or custom query
Event table name	EventTableName	Name of the database table that contains events generated by the database for inbound processing
Event types to process	EventTypeFilter	A delimited list of event types that indicates to the adapter which events it should deliver
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
Number of times to retry the system connection	RetryLimit	The number of times the adapter tries to reestablish an inbound connection after an error

Table 78. Activation specification properties for Adapter for Oracle E-Business Suite (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
Password	Password	Password for authorizing the user to retrieve events from the database
Poll quantity	PollQuantity	The number of events that the adapter delivers to the export during each poll period
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Retry EIS connection on startup	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the Oracle database if it cannot connect at startup
Retry interval if connection fails	RetryInterval	The length of time that the adapter waits between attempts to establish a new connection after an error during inbound operations
Return dummy business object for RetrieveSP	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
SQL query to verify the connection	PingQuery	The SQL query used to test 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

Adapter instance for event filtering (AdapterInstanceEventFilter)

This property controls whether this adapter instance processes specific events in the event store.

Table 79. Adapter instance for event filtering details

Required	No
Default	`null`
Property type	String

Table 79. Adapter instance for event filtering details (continued)

Usage	This property helps you migrate from WebSphere Business Integration Adapter for Oracle Applications to WebSphere Adapter for Oracle E-Business Suite. WebSphere Business Integration Adapter for Oracle Applications 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 Oracle E-Business Suite 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 Oracle Applications. To use this feature, the database triggers or other mechanisms that create events in the event store must assign the appropriate value to the connectorId column. Table 80 shows the interaction between the AdapterInstanceEventFilter property and the value in the connectorId column in the event store. If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose connectorId column matches the AdapterInstanceEventFilter property.
Example	See Table 80.
Globalized	Yes
Bidi supported	Yes

Table 80. Interaction of the AdapterInstanceEventFilter property with the connectorId column in the event store

AdapterInstanceEventFilter property	**connectorId column of an event**	**Result**
`null`	`null`	The adapter processes the event
`null`	`Instance1`	The adapter processes the event, because the connectorId 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 (DriverConnectionProperties)

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

Table 81. Additional JDBC driver connection properties details

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

Table 81. Additional JDBC driver connection properties details (continued)

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: `loginTimeout:20; ConnectionRetryCount:5; ConnectionRetryDelay:5`
Globalized	Yes
Bidi supported	No

Business object namespace (BusinessObjectNameSpace)

This property specifies the namespace for the business object definitions.

Table 82. Business object namespace property characteristics

Required	No
Default	`http://www.ibm.com/xmlns/prod/websphere/j2ca/oracle`
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/oracle/Schema1Customer`
Bidi supported	No

Custom delete query (CustomDeleteQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run after each event is processed to delete records that can be deleted after the event is delivered.

Table 83. Custom delete query details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the SQL statement, stored procedure, or stored function to run when the EventQueryType property is set to `Dynamic`.
Globalized	Yes
Bidi supported	Yes

Custom event query (CustomEventQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run to poll for events in custom event processing.

Table 84. Custom event query details

Required	No
Default	No default value

Table 84. Custom event query details (continued)

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 an SQL statement that returns the event ID, object key, and object name of every record in the MY_EVENT_TABLE event store whose status column has the value `0`:
	`select event_id, object_key, object_name from MY_EVENT_TABLE where status = 0`
	The following example limits the returned event records to the value of the PollQuantity property:
	`select event_id, object_key, object_name from MY_EVENT_TABLEwhere status = 0`
	`and rownum < POLL QUANTITY`
	The following example runs a stored procedure with two parameters:
	`CALL MY_EVENT_STORED_PROC (?,?)`
	The following example runs a stored function with one parameter and one return value:
	`? = CALL MY_EVENT_FUNCTION(?)`

Globalized	Yes
Bidi supported	Yes

Custom update query (CustomUpdateQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run after each event is processed so that the same event does not get picked up for processing in the subsequent event cycle.

Table 85. Custom update query details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the SQL statement, stored procedure, or stored function to run when the EventQueryType property is set to `Dynamic`.
Globalized	Yes
Bidi supported	Yes

Data source JNDI name (DataSourceJNDIName)

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

Table 86. Data source JNDI name details

Required	No
Default	No default value
Property type	String

Table 86. Data source JNDI name details (continued)

Usage	Use this property to specify the JNDI name of a data source in WebSphere Application Server 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. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName property, if set, to establish the connection. The DataSourceJNDIName property represents an XA or connection pool data source. If you define a JNDI data source on the server that supports XA transactions and then specify that data source when you configure the adapter, the adapter participates in XA transactions. Optionally, you can specify the XADataSourceName property for the adapter to participate in XA transactions. 3. If the DataSourceJNDIName and XADataSourceName properties are not set, the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. Do not confuse the data source JNDI name property with the JNDI name of a managed connection factory or activation specification on the server. The following list highlights important differences between the types of JNDI names: • Data source JNDI name – Specifies a connection to a database – 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 87. Database URL details

Required	Yes, unless any of the following properties or sets or properties are set: • The DataSourceJNDIName • The XADataSourceName property
Default	No default value

Table 87. Database URL details (continued)

Property type	String
Usage	In the J2C Bean wizard, compose the database URL by filling in database-specific fields. For example, the database URL for an Oracle database is composed of the system ID (SID), server host name, and database port number. In the administrative console, type the complete database URL value. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName property, if set, to establish the connection. The DataSourceJNDIName property represents an XA or connection pool data source. If you define a JNDI data source on the server that supports XA transactions and then specify that data source when you configure the adapter, the adapter participates in XA transactions. Optionally, you can specify the XADataSourceName property for the adapter to participate in XA transactions. 3. If the DataSourceJNDIName and XADataSourceName properties are not set, the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets ([]).
Examples	The following example illustrates typical database URL values for a common database:`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 88. Database vendor details

Required	Yes
Possible values	`Oracle`
Default	`ORACLE`
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. Specify `Oracle` as the value that corresponds to your database vendor.
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 89. Delivery type details

Required	No
Possible values	`ORDERED` `UNORDERED`
Default	`ORDERED`
Property type	String
Usage	The following values are supported: • `ORDERED`: The adapter delivers events to the export one at a time. • `UNORDERED`: The adapter delivers all events to the export at once.
Globalized	No
Bidi supported	No

Do not process events that have a timestamp in the future (FilterFutureEvents)

This property specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time.

Table 90. 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 91. Ensure once-only event delivery details

Required	Yes
Possible values	`True` `False`
Default	`True`
Property type	Boolean

Table 91. 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 92. 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 (EventQueryType)

This property specifies whether to use standard or custom query processing.

Table 93. Event query type details

Required	Yes
Possible values	`Standard` `Dynamic`
Default	`Standard`
Property type	String
Usage	The valid values are `Standard`, for standard event processing, and `Dynamic`, for custom event processing. If this property is set to `Dynamic`, the CustomEventQuery, CustomUpdateQuery, and CustomDeleteQuery properties are used. If this property is set to `Standard`, those properties are ignored.
Globalized	No

Table 93. Event query type details (continued)

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 94. Event table name details

Required	Yes
Default	`WBIA_Oracle_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 95. 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.
Example	To receive only events relating to the Customer and Order business objects, specify this value: `Customer,Order` If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose connectorId column matches the AdapterInstanceEventFilter property.
Globalized	No
Bidi supported	No

Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 96. Retry limit for failed events details

Required	No

Table 96. Retry limit for failed events details (continued)

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 97. JDBC driver class details

Required	Yes, if the DataSourceJNDIName property is not set
Possible values	The value specified when you set the connection properties to connect to the database instance.
Default	No default value
Property type	String
Usage	In the J2C Bean wizard, the JDBC driver class is specified for you if you select a common database software and driver combination, such as type 4 drivers for recent versions of Oracle. For type 2 drivers , you must type the database class name. If you select a type 2 driver or a generic driver, you must type the JDBC driver class name. In the administrative console, type the database-specific name of the driver. If the DataSourceJNDIName property is set, this property is ignored.
Examples	Values for the JDBC driver class display in both the J2C Bean wizard and the administrative console. The following examples depict the JDBC driver class properties in both the J2C Bean wizard and the administrative console.**In the J2C Bean wizard:** • To connect to an Oracle 10 database using the type 4 driver, select `Oracle Thin Driver`. **In the administrative console:** **Oracle Thin JDBC driver** `oracle.jdbc.driver.OracleDriver`
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 98. Maximum connections details

Required	No
Default	1
Property type	Integer
Usage	Only positive values are valid. The adapter considers any positive entry less than 1 to be equal to 1. Typing a negative value or 1 for this property may result in runtime 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 99. 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 100. Password details

Required	No. If you set the Authentication alias or the DataSourceJNDIName, the password is not mandatory. However, if you set the Authentication alias, the DataSourceJNDIName, and Password field, the value specified for the Password takes precedence.
Default	No default value
Property type	String
Usage	If set, this property overrides the password specified on a data source on the server using the Authentication alias or DataSourceJNDIName 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 101. 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

Interval between polling periods (PollPeriod)

This property specifies the length of time that the adapter waits between polling periods.

Table 102. 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 103. 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 may improve slightly.
Globalized	No
Bidi supported	No

Query timeout (QueryTimeOut)

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

Table 104. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Retry interval if connection fails (RetryInterval)

When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.

Table 105. Retry interval details

Required	Yes
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.
Globalized	Yes
Bidi supported	No

Number of times to retry the system connection (RetryLimit)

This property specifies the number of times the adapter tries to reestablish an inbound connection.

Table 106. 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 Oracle 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 Oracle database when it is first started, use the RetryConnectionOnStartup property.

Table 106. Number of times to retry the system connection details (continued)

Globalized	Yes
Bidi supported	No

Retry EIS connection on startup (RetryConnectionOnStartup)

This property controls whether the adapter attempts to connect again to the Oracle database if it cannot connect at startup.

Table 107. Retry EIS connection on startup details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	This property indicates whether the adapter should retry the connection to the Oracle 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 Oracle 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 108. 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 populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes

Table 108. Return business object even when the stored procedure result set is empty details (continued)

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 109. 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 110. Stored procedure to run after poll details

Required	No
Default	No default value
Property type	String
Usage	The stored procedure takes one parameter for poll quantity.
Globalized	Yes
Bidi supported	Yes

Stored procedure to run before polling (SPBeforePoll)

This property specifies the name of any stored procedure or stored function to run before the actual poll query is called.

Table 111. Stored procedure to run before poll details

Required	No
Default	No default value
Property type	String
Usage	The stored procedure takes one parameter for poll quantity.
Globalized	Yes
Bidi supported	Yes

User name (UserName)

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

Table 112. User name details

Required	No. If you set the Authentication alias or the DataSourceJNDIName, the User name property is not mandatory. However, if you set the Authentication alias, the DataSourceJNDIName, and User name, the value specified for the User name takes precedence.
Default	No default value
Property type	String
Usage	Setting this property overrides the user name specified on a data source on the server using the DataSourceJNDIName property or the Authentication alias.
Globalized	Yes
Bidi supported	Yes

Globalization

WebSphere Adapter for Oracle E-Business Suite 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 left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file.

Globalization

Globalized software applications are designed and developed for use within multiple linguistic and cultural environments rather than a single environment. WebSphere Adapters, Rational Application Developer for WebSphere Software, and WebSphere Application Server 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 Application Server uses the

Windows® standard format, but an enterprise information system exchanging data with WebSphere Application Server can use a different format. WebSphere Adapters transform bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction.

Bidirectional format

WebSphere Application Server uses the bidirectional format of ILYNN (implicit, left-to-right, on, off, nominal). This is the format used by Windows. If an enterprise information system uses a different format, the adapter converts the format prior to introducing the data to WebSphere Application Server.

Five attributes comprise bidirectional format. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 113. Bidirectional format attributes

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

The adapter transforms data into a logical, left-to-right format before sending the data to WebSphere Application Server.

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 114. 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 J2C Bean 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 Oracle E-Business Suite 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 J2C Bean 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.

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

Connection properties used in the wizard

The following connection properties for the J2C Bean 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

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

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning:

Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

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

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

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

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org).

Index

Printed in USA