

WebSphere Adapters
Version 7 Release 0 Feature Pack 1

*WebSphere Adapter for JDBC User
Guide
Version 7 Release 0 Feature Pack 1*

IBM

WebSphere Adapters
Version 7 Release 0 Feature Pack 1

*WebSphere Adapter for JDBC User
Guide
Version 7 Release 0 Feature Pack 1*

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 131.

June 2010

This edition applies to version 7, release 0, modification 1 of IBM WebSphere Adapter for JDBC and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

© **Copyright IBM Corporation 2006, 2010.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. What is new in this release. . . . 1

Chapter 2. User-defined types in Oracle table and query business objects. . . . 3

Business objects	3
Create operation.	15
Update operation	16
Selecting and configuring tables, views, and synonyms or nicknames for outbound processing	17
Selecting and configuring query business objects	25
Business object attributes	29

Chapter 3. XML data type in table business objects 33

Adding external software dependencies	33
Business object attributes	35
Attribute application-specific information	38
Solutions to common problems.	45

Chapter 4. Customize format for date, time, and timestamp data types 69

Business objects	69
----------------------------	----

Discovering database objects for outbound processing.	81
Discovering database objects for inbound processing	85
Attribute application-specific information	87

Chapter 5. Automatic discovery of parent-child relationship in tables 95

Business objects	95
Discovering database objects for outbound processing	107
Discovering database objects for inbound processing	111
Selecting and configuring tables, views, and synonyms or nicknames for outbound processing	113
Selecting and configuring tables, views, and synonyms or nicknames for inbound processing.	121

Notices 131

Programming interface information	133
Trademarks and service marks	133

Chapter 1. What is new in this release

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

Complete information about other supported features is available at the WebSphere® Adapter for JDBC information center, http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.wsadapters.jca.jdbc.doc/doc/stbp_jdb_welcome.html, which is periodically updated with the latest information.

WebSphere Adapter for JDBC supports the following new features:

- User-defined types in Oracle table and query business objects.
- XML data type in table business objects.
- Customize format for date, time, and timestamp data types.
- Automatic discovery of parent-child relationship in tables.

Note: In WebSphere Integration Developer, ensure that you have only one version of the adapter imported into your workspace. You can either have the adapter Fix Pack version 7.0.0.1 or Feature Pack version 7.0.1.0.

In the runtime environment, the application (EAR) should contain only one version of the embedded RAR file, either the adapter Fix Pack version 7.0.0.1 or Feature Pack version 7.0.1.0. The node level deployed adapter should also have only any one version of the adapter.

Chapter 2. User-defined types in Oracle table and query business objects

The adapter supports user-defined or complex data types such as array, table, structure, or nested structure in table and query business objects for Oracle database. The type name and child attribute details are automatically discovered and displayed for these types.

If you have columns defined as user-defined types, the external service wizard automatically discovers these columns and generates child business object for the user-defined type. The adapter transforms the input business object instance to user-defined type instance and uses the user-defined type instance to perform corresponding operation in outbound processing.

Business objects

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

How the adapter uses business objects

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

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

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

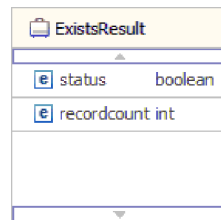
Some business objects do not represent database objects. These business objects include:

- Batch SQL business objects, which represent a series of user-defined insert, update, and delete statements.
- Query business objects, which represent a user-defined SQL query to run against the database.
- Wrapper business objects, which allow you group unrelated table and view objects into a single business object, and multiple stored procedures into a single business object.

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

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

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



How data is represented in business objects

For table or view business objects

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

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

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

A table business object always has a primary key, even if the corresponding database table does not have a primary key. The adapter uses the column specified in the primary key attribute when it retrieves table business objects. If you have defined foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables. For example, consider tables CUSTOMER and ADDRESS, where CUSTOMER is the parent table and ADDRESS is the child table. If you have defined a foreign key reference from ADDRESS to CUSTOMER in the database, the adapter automatically discovers the parent-child relationship displays the foreign key reference in the Specify the Configuration Properties for 'object' window. If the foreign key reference is from CUSTOMER to ADDRESS, the adapter automatically selects the **Single cardinality** check box and displays the foreign key reference. If there are multiple foreign key references defined for a table, the adapter generates only one foreign key relationship.

The adapter supports tables that have composite, or multiple primary keys. If a database table has one or more primary keys, the wizard sets the primary key property for those columns in the table business object. If the database table does not have a primary key, the external service wizard prompts you for primary key

information when you discover and configure that business object. If there is a composite primary key reference, for instance CUSTOMER (pkey1, pkey2) > ADDRESS (fkey1, fkey2), the adapter associates ADDRESS (fkey1) with CUSTOMER (pkey1) and ADDRESS (fkey2) with CUSTOMER (pkey2). Specify a column that contains unique data, such as a sequence or identity column. Identity columns (known as *serial columns* in Informix®; JDBC adapter supports both serial and serial8) provide a way for the database to automatically generate a unique numeric value for each row in a table. A table can have a single column that is defined with the identity attribute. Examples of an identity column include order number, employee number, stock number, and incident number. Identity columns can be defined for tables in DB2®, Informix and Microsoft® SQL Server only.

Note: When you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object for that table does not include the Unique Identifier attribute of the identity column. In this case, you need to edit the generated business object by adding the attribute to the application-specific information manually. You can do this through the assembly editor in WebSphere Integration Developer. You do not need to add the attribute for the Unique Identifier manually if you ran the discovery process against a table in an Informix database. For Informix, the generated business object includes the Unique Identifier attribute of the serial column.

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.fffffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

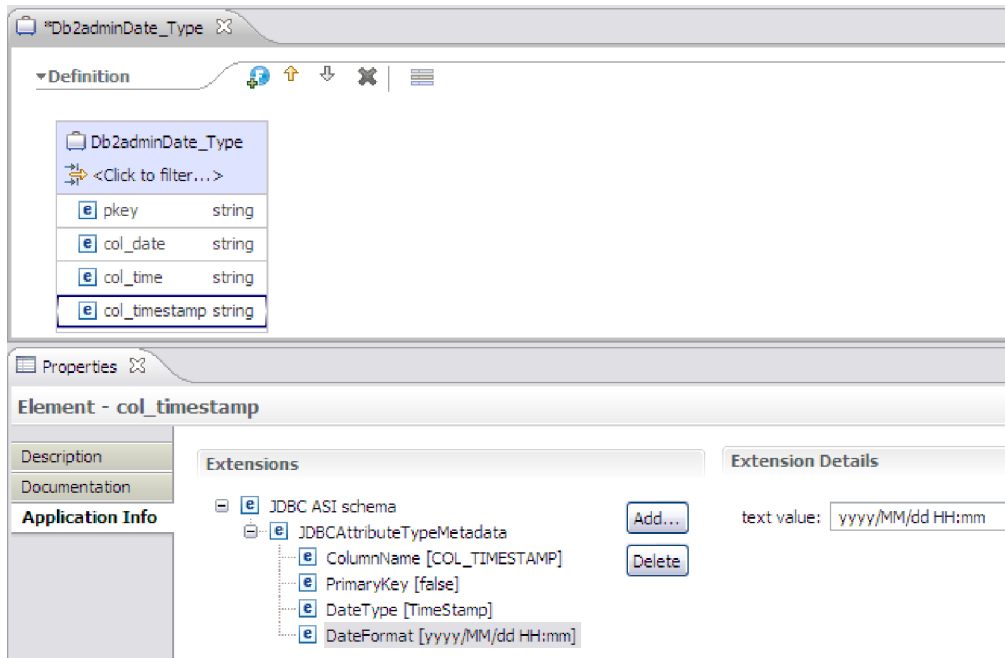


Figure 1. The DateFormat application-specific information with the customized format

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

Figure 2 shows a table business object in the business object editor. The business object has an attribute for each of the columns in the database table. Because the table has no child business objects, all of the attributes are simple attributes.

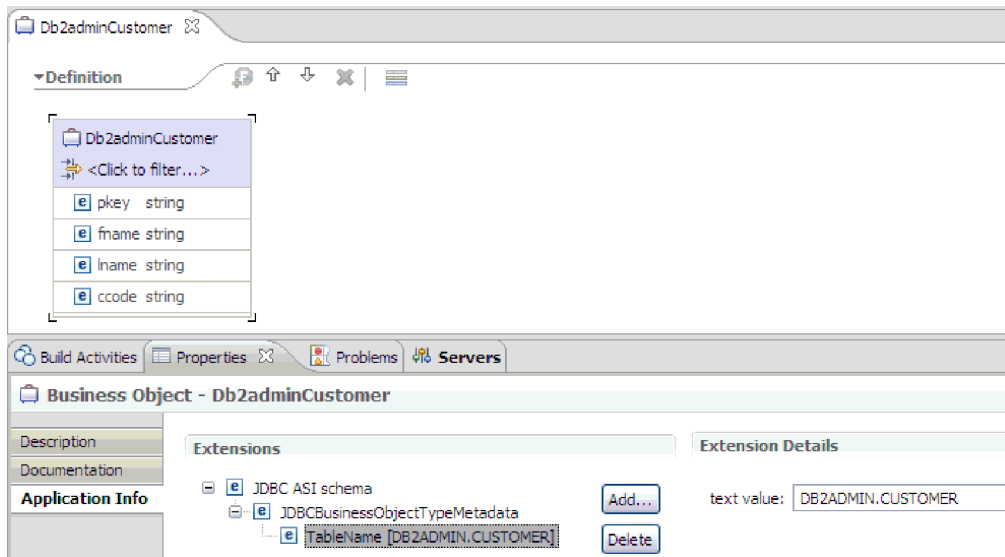


Figure 2. A table business object with no child.

Figure 3 on page 7 shows a table business object that has one child table business object. The business object has simple attributes for each of the columns in the

database table, plus a complex attribute pointing to a child business object.

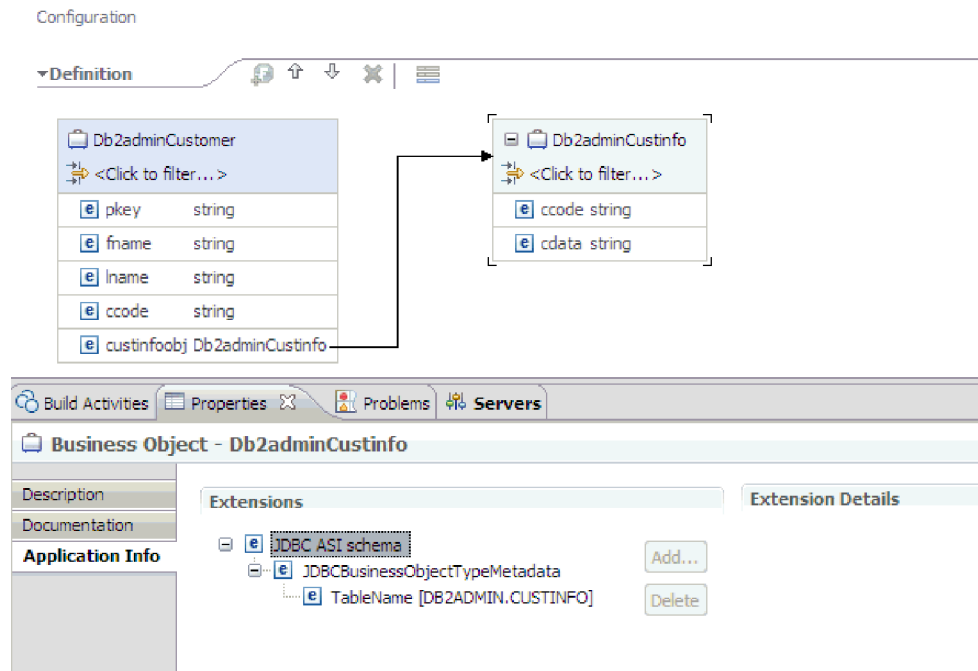


Figure 3. A table business object with one child business object.

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

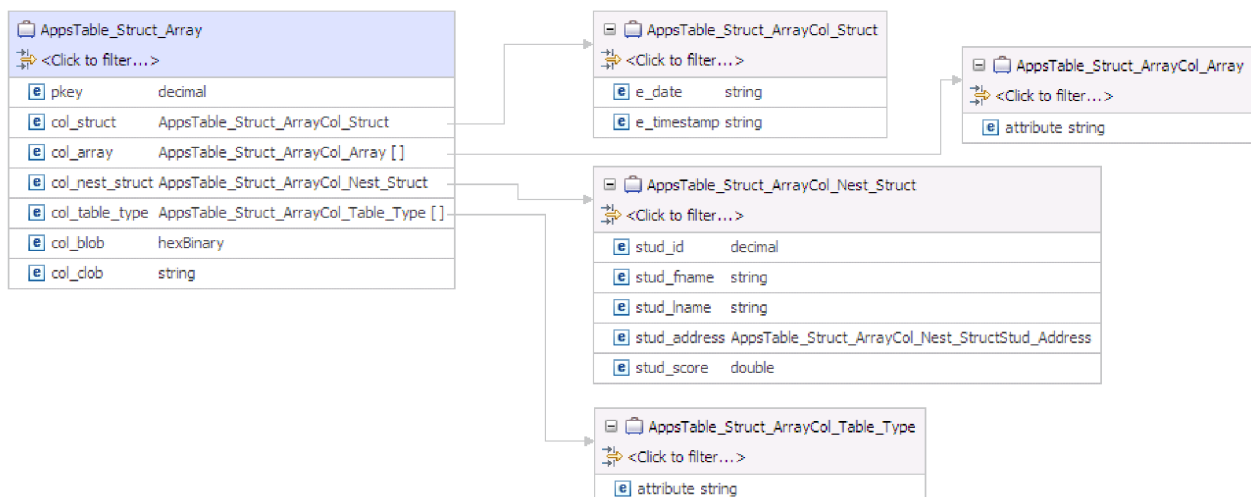


Figure 4. An Oracle table business object having user-defined or complex types as columns

For stored procedure and stored function business objects

In a business object for a stored procedure or stored function, all the input and output parameters for the stored procedure or stored function have corresponding

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

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.ffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

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

Example of ScottStrtValues business object

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
```

```

<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>IP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
<jdbcasi:ChildBOTType>STRUCT</jdbcasi:ChildBOTType>
<jdbcasi:ChildBOTTypeName>STRUCT1</jdbcasi:ChildBOTTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

Example of ScottStrtValuesStrt business object

```

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asi:SURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>

```

```

</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="name" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="title" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

For query business objects

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

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

```

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

```

The question mark (?) indicates an input parameter for the query. A query can have multiple parameters, each indicated in the SELECT statement by a question mark. Table 1 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 1. 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

Table 1. Attributes of a query business object (continued)

Business object attribute	Description
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.
jdbewhereclause	A placeholder object for the WHERE clause

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is `"yyyy-MM-dd"`, Timestamp type is `"yyyy-mm-dd hh:mm:ss.fffffffff"`, and Time type is `"HH:mm:ss"`.

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

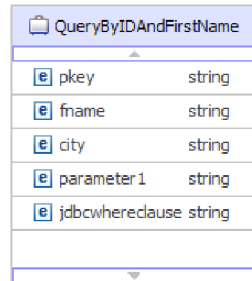


Figure 5. The attributes of a query business object

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

statement.

For Oracle databases, the adapter supports complex data types such as array, table,

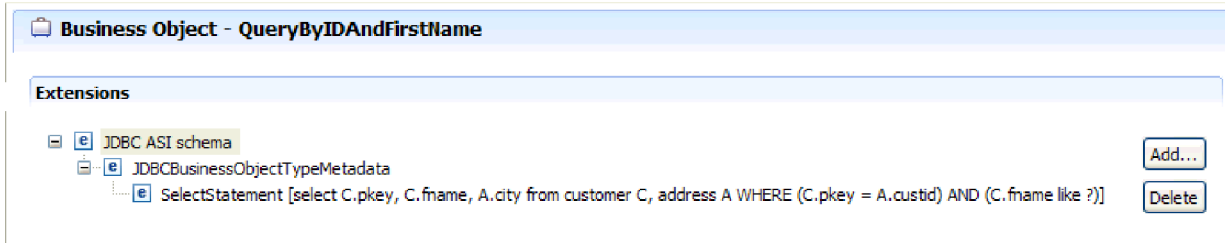


Figure 6. The SELECT statement is saved in the business object application-specific information

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

For batch SQL business objects

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

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

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

Each question mark (?) indicates a parameter for the statement. Each statement in a batch SQL business object can have multiple parameters, each indicated in the statement by a question mark. A batch SQL business object can have multiple statements, each with its own set of parameters. Figure 7 shows the format of the business object for the batch SQL business object with an INSERT and a DELETE statement, each of which has one or more parameters.

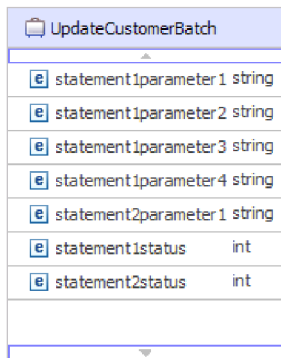


Figure 7. A batch SQL business object with two SQL statements

The business object has an attribute for each parameter in each statement, including statement1parameter1, statement2parameter1, and so on. It also has an attribute for the status of each statement, such as statement1status, statement2status, and so on. The statements themselves are stored as application-specific information about the business object, as shown in Figure 8 on page 13

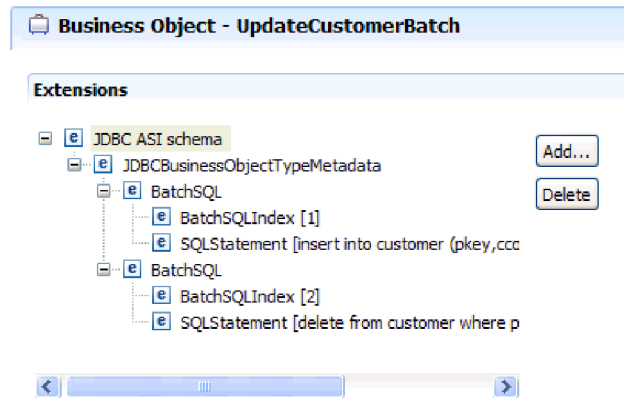


Figure 8. The application-specific information of a batch SQL business object

For wrapper business objects

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

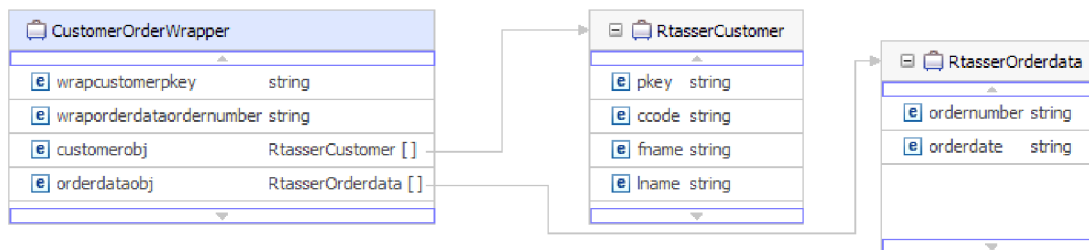


Figure 9. A wrapper business object that contains two table business objects

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

Business graphs

You can optionally choose, during adapter configuration, to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use in version 6.0.2 to specify additional information about the operation to be performed. In version 7.0, business graphs are required only in these situations:

- If you need to use the outbound ApplyChanges operation
- When adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 7.0

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

How business objects are created

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

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

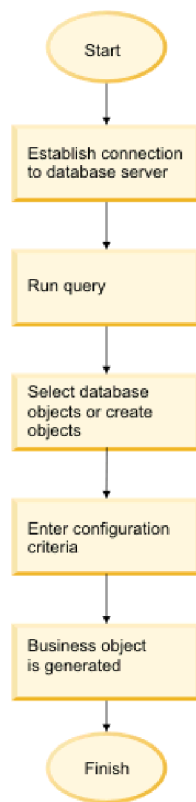


Figure 10. How business objects are created

In some instances, the wizard cannot completely configure a parent-child relationship. For these relationships, you use the business objects editor, launched from WebSphere Integration Developer, to modify or complete the definition of a business object hierarchy that was created by the wizard. For more information, see the instructions for using the business object editor to modify business objects in the WebSphere Integration Developer information center at the following link: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp>

Create operation

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

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

1. Checks whether the business object is a wrapper. If the top-level business object is a wrapper business object, the adapter ignores the business object. No rows are created for wrapper objects.
2. Recursively inserts each single-cardinality child business object contained with ownership into the database. In other words, the adapter creates the child and all child business objects that the child and its children contain.

If the business object definition specifies that an attribute represents a child business object with single-cardinality and that attribute is empty, the adapter ignores the attribute. However, if the business object definition requires that the attribute represent a child, and it does not, the adapter returns an error and stops processing.

3. Retrieves and checks for the existence of each single-cardinality child business object contained without ownership. If the retrieval is unsuccessful, indicating that the child does not exist in the database, the adapter returns an error, and stops processing. If the Retrieve operation is successful, the adapter recursively updates the child business object.

Note: For this approach to work correctly when the child business object exists in the database, primary-key attributes in child business objects must be cross-referenced correctly on Create operations. If the child business object does not exist in the application database, the primary-key attributes must not be set.

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

Note: The adapter treats an empty complex column as a null column irrespective of setting the value to null or unset.

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

- b. Inserts each of the multiple-cardinality child business objects into the database.

Update operation

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

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

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

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

If the retrieval succeeds, the adapter compares the retrieved business object to the source business object to determine which child business objects require changes in the database. The adapter does not, however, compare values in the source business object's simple attributes to those in the retrieved business object. The adapter updates the values of all non-key simple attributes.

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

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

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

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

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

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

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

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

If the top-level business object is a wrapper business object, it is ignored. No update is done for wrapper business objects.

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

NULL data and the Update operation

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

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

Note: The adapter treats an empty complex column as a null column irrespective of setting the value to null or unset.

Selecting and configuring tables, views, and synonyms or nicknames for outbound processing

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

Before you begin

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

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

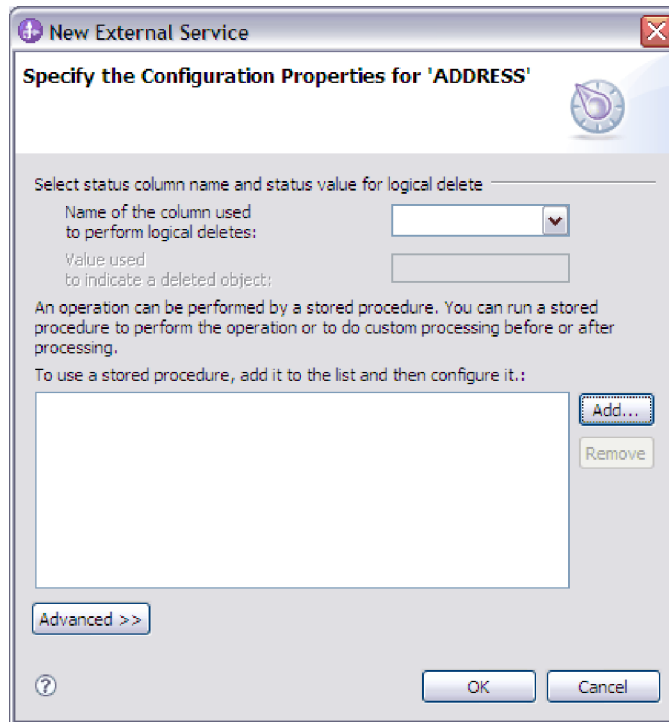
About this task

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

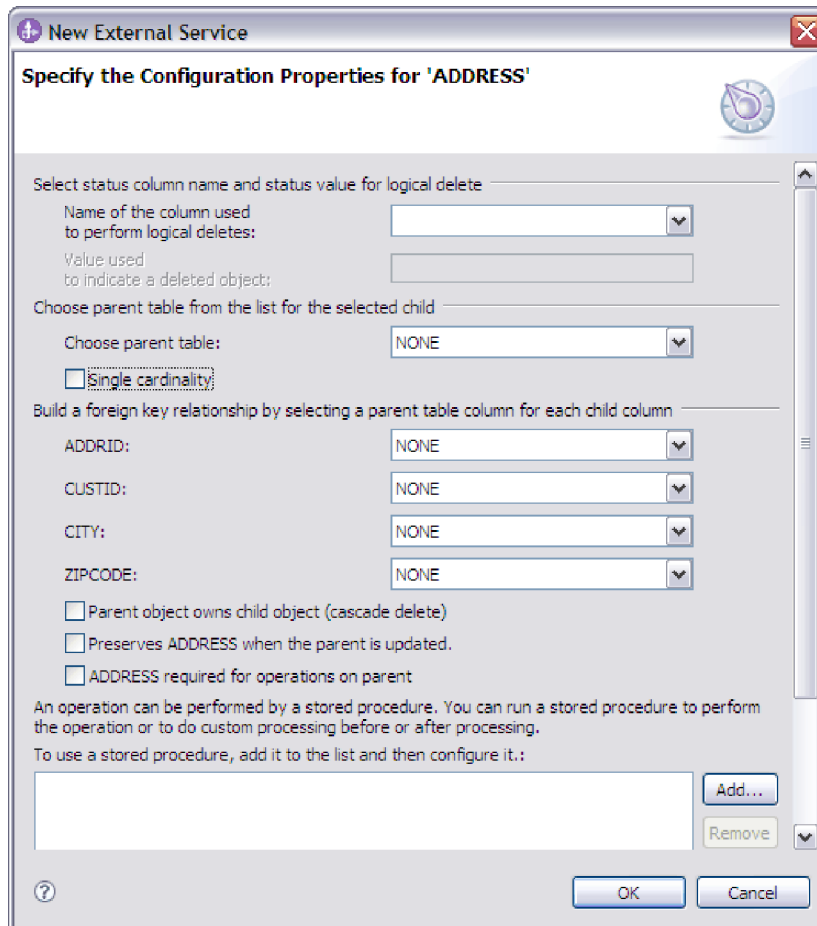
Procedure

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

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



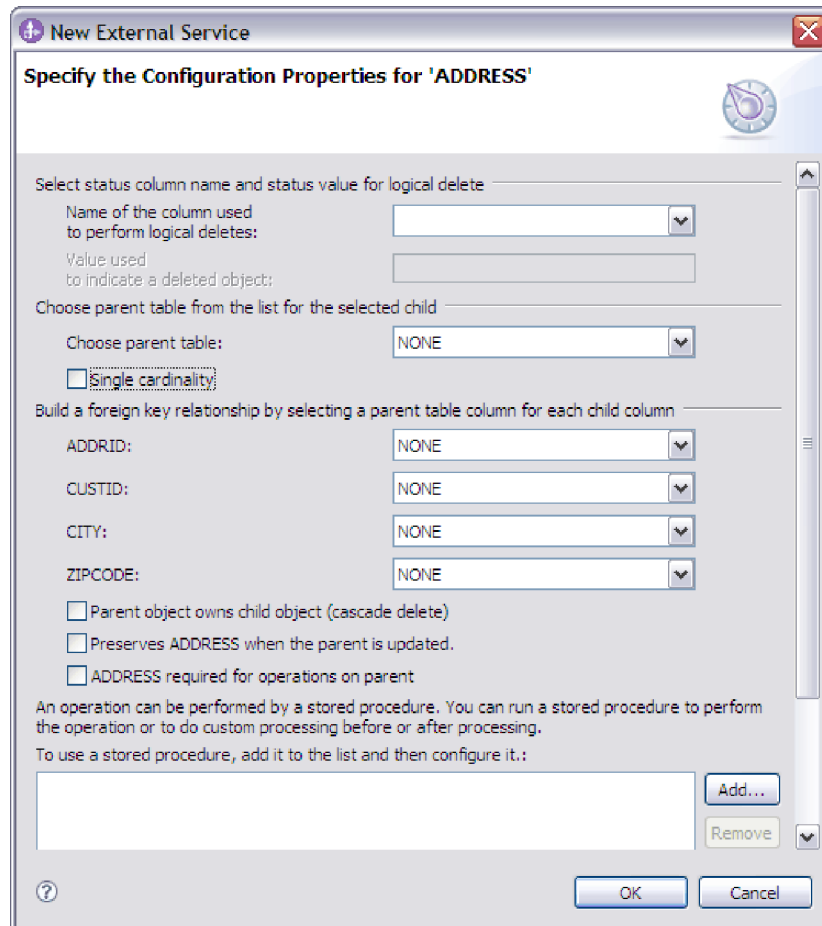
The following figure shows a typical window for subsequent tables you select. After you select and configure at least one table, the Specify the Configuration Properties for 'object' window for subsequent tables displays an area where you can optionally define a parent-child hierarchy between tables.



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

2. If the table has a column that is used to indicate logical deletes:
 - a. Select the column name in the **Name of the column used to perform logical deletes** field.
 - b. In the **Value used to indicate a deleted object** field, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. If the **Select primary key for table *table_name*** area is displayed, click **Add**, select the column to be used as the primary key for the table business object, and then click **OK**. If the table has a composite key, you can select multiple columns. The **Select primary key for table *table_name*** area is displayed only when the database table does not have a column designated as the primary key. Each table business object must have a primary key, even if the associated database table does not have a key. If the primary key is defined in the database, this section of the window is not displayed.
4. Optional: Define a parent-child relationship between business objects.
To build a parent-child hierarchy, configure the parent table first, and return to the Find Objects in the Enterprise System window to select and configure the child tables.

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



- a. In the **Choose parent table** field, select the name of the parent table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects. If you have defined a foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables after you select the parent table. If the table has a single-cardinality relationship with the parent table, the **Single cardinality** check box is automatically selected.
- b. Specify the cardinality of the relationship:
 - If the table has a single-cardinality relationship with the parent table, select the **Single cardinality** check box. In a single cardinality relationship, a parent can have only one child business object of this type. A single-cardinality relationship can be used with ownership to represent a true child or without ownership to represent lookup tables or other peer objects in a database.
 - If the table has a multiple-cardinality relationship, do not select the **Single cardinality** check box. In a multiple-cardinality relationship, a parent can have an array of child business objects of this type.
- c. Build the foreign key relationship between the parent and child by specifying for each child column whether it is a foreign key in the parent table.
 - If the child column is not a foreign key, select NONE.
 - If a child column is a foreign key, select the column in the parent table that corresponds to the child column.

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

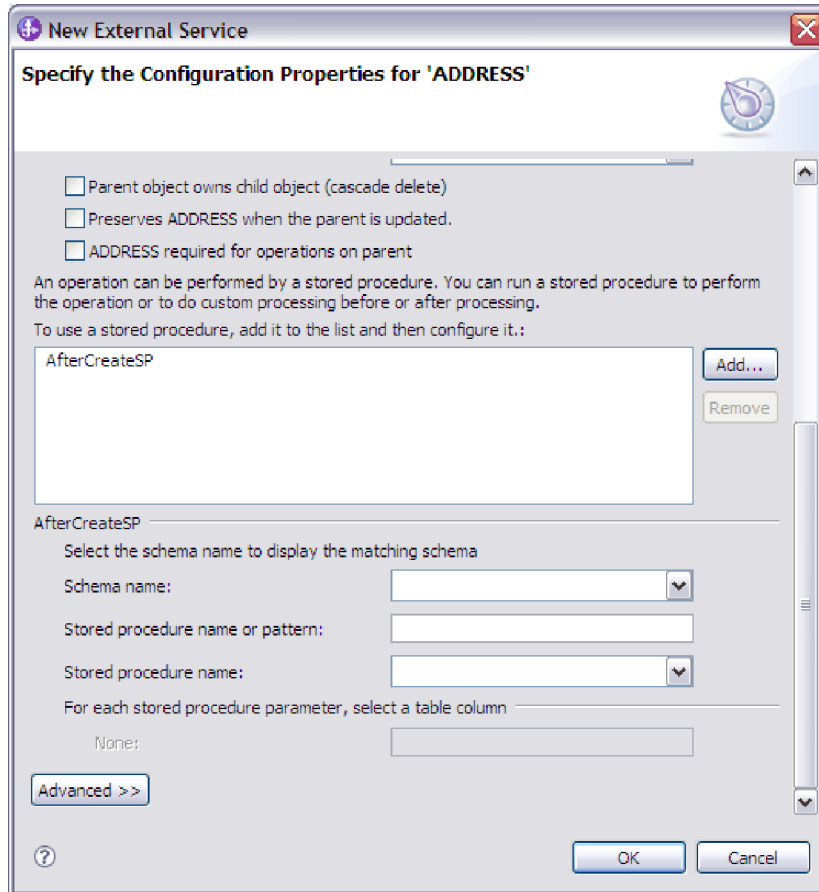
- d. If the parent object owns the child object, then the child objects in the database are deleted when the parent is deleted. To indicate that this child is owned by its parent, select the **Parent object owns child object (cascade delete)** check box. Otherwise, clear this option to prevent child objects, such as lookup tables, from being deleted when their parent is deleted.
 - e. If you do not want child objects to be deleted as part of an Update operation, select the **Preserves *child_table_name* when the parent is updated** check box.

When a parent table is updated, the adapter compares the child business objects present in the input with the child business objects returned from the database. By default, the adapter deletes any child objects returned from the database that are not present in the input business object.
 - f. By default, you can perform operations on parent business objects without specifying the child business objects. If you want to ensure that a parent business object specifies its child business objects when the parent is submitted for a change, select the ***Child_table_name* required for operations on parent** check box.
5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

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

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

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

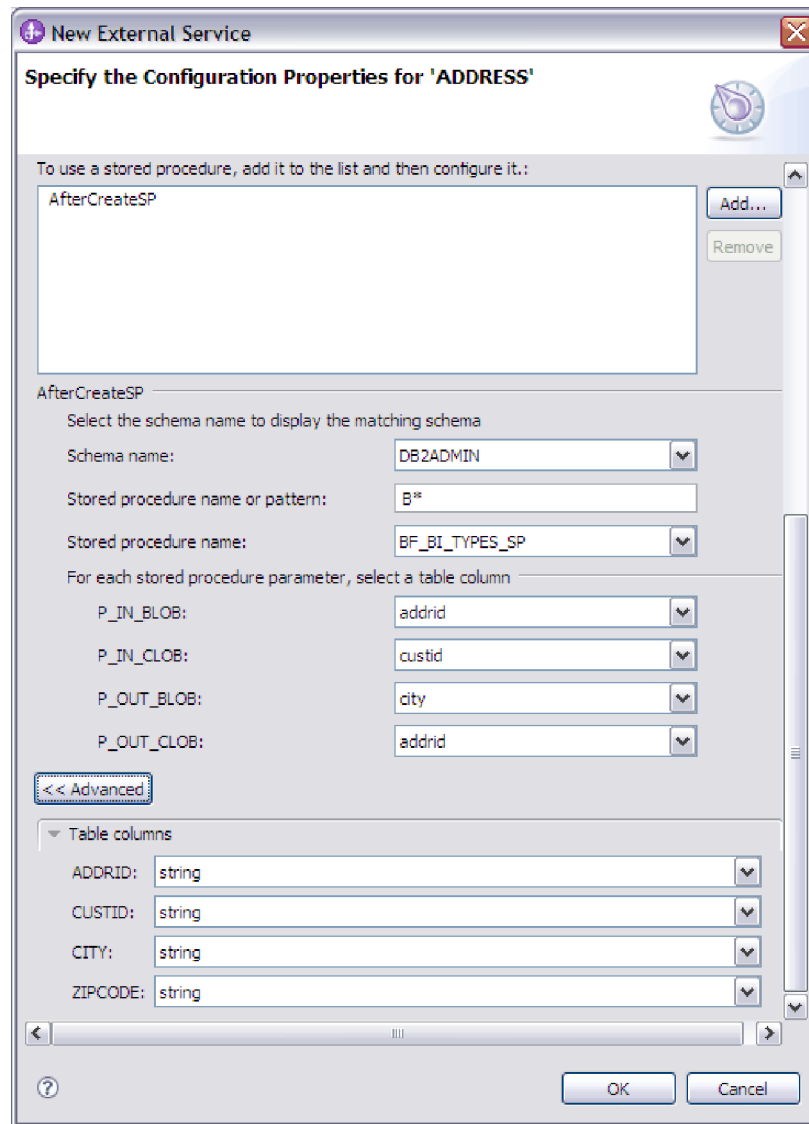


Note: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored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

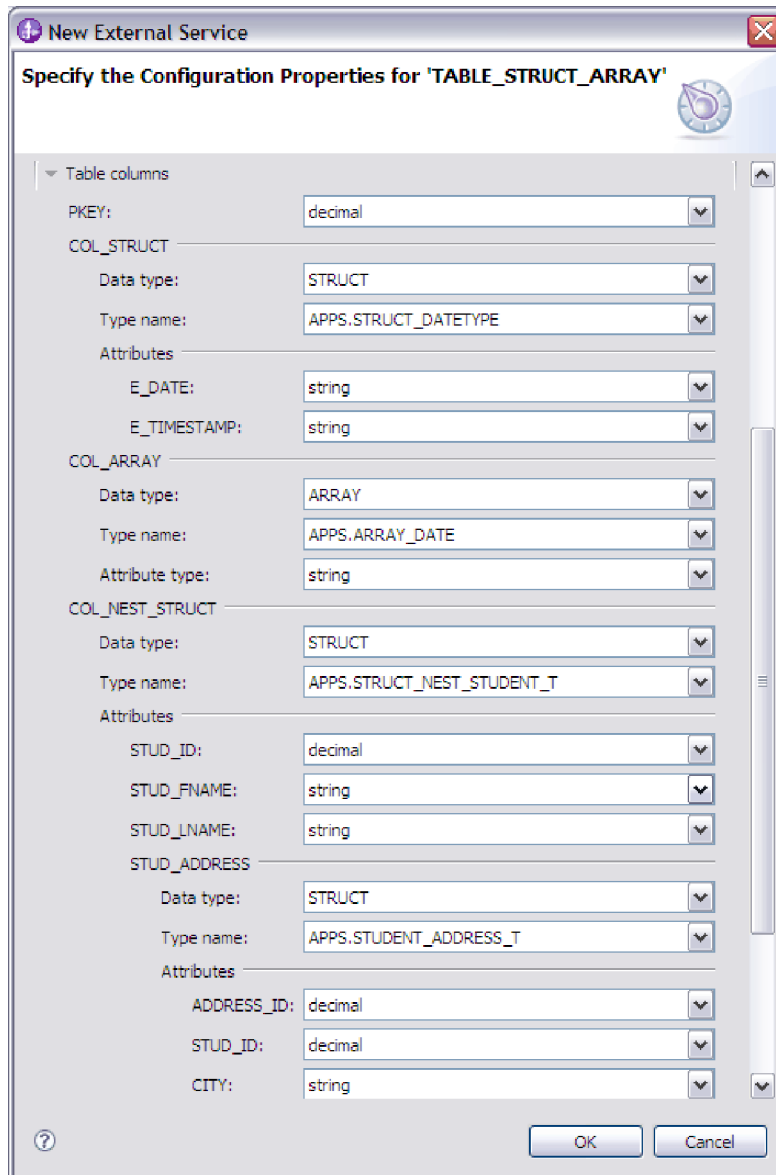
6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
 - a. In the **Schema name** field, select the name of the schema that contains the stored procedure.
 - b. Specify the name of the stored procedure or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


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

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



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



- c. Review the mapping and make changes if required.
8. When all fields in the window are completed, click **OK** to save the configuration of the business object. The table, view, synonym, and nickname business objects you defined are now listed in the Find Objects in the Enterprise System window.
9. To change the configuration of an object from the **Selected objects** list, select the object name and then click the  (Edit) icon.

What to do next

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

Selecting and configuring query business objects

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

Before you begin

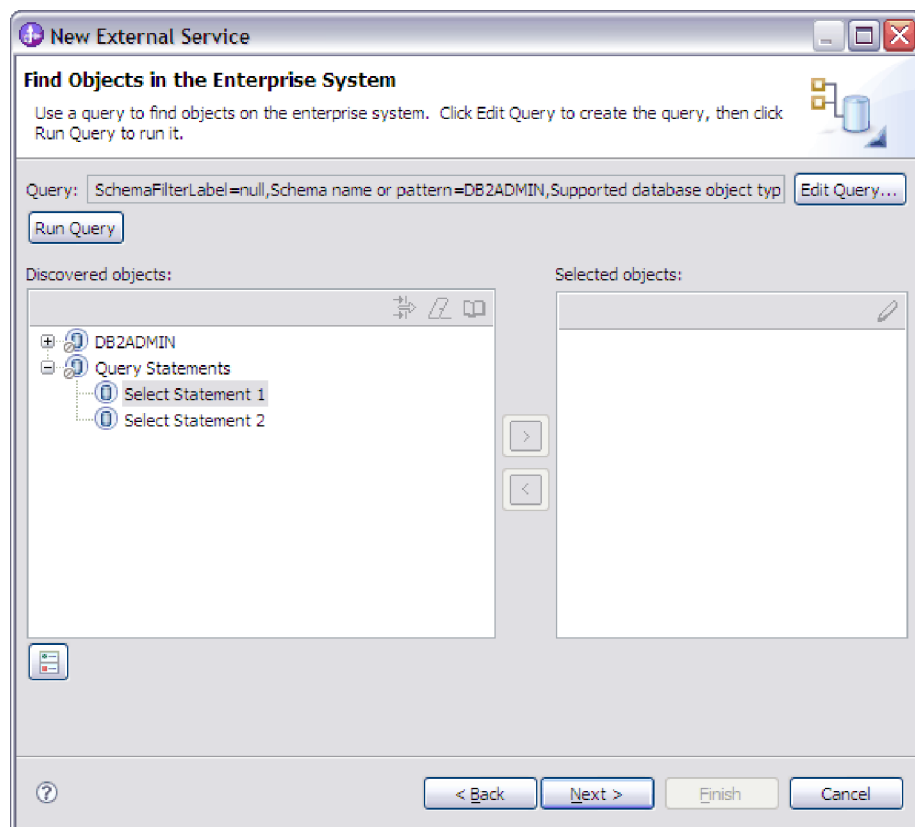
To configure query business objects, you must know the structure of the data in your database, including the tables and views. You must know the name and data type of the columns that your module must access. You must also be able to write SQL SELECT statements.

About this task

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

Procedure

1. In the **Discovered objects** list of the Find Objects in the Enterprise System window, expand the **Query Statements** node. This node contains an object template, named **Select Statement *n***, for each query business object you requested in the Specify the Query Properties window. For example, if you specified a count of two query business objects in that window, the **Discovered objects** list contains two object templates, as illustrated in the figure below:

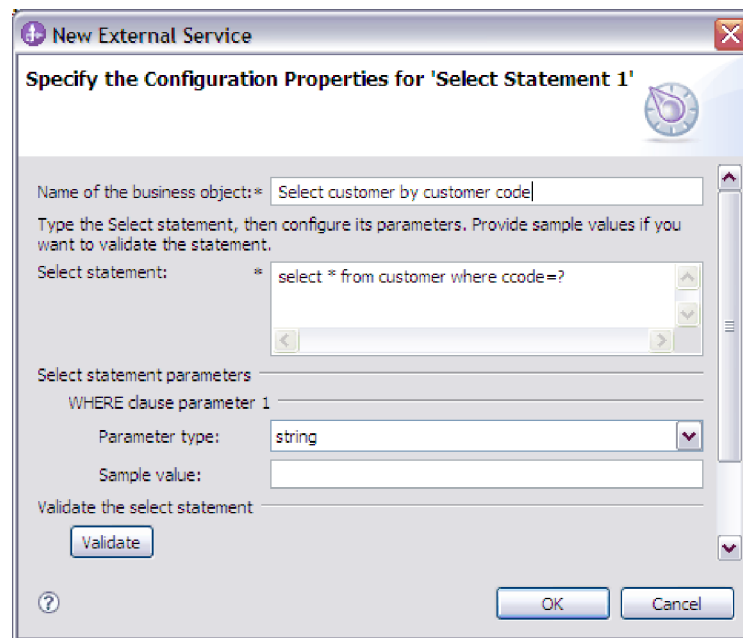


2. Select one or more object templates and click the > (Add) button to add the objects to the **Selected objects** list.

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

Note: Ensure that the SELECT statement does not include a nested SELECT statement in the FROM clause.

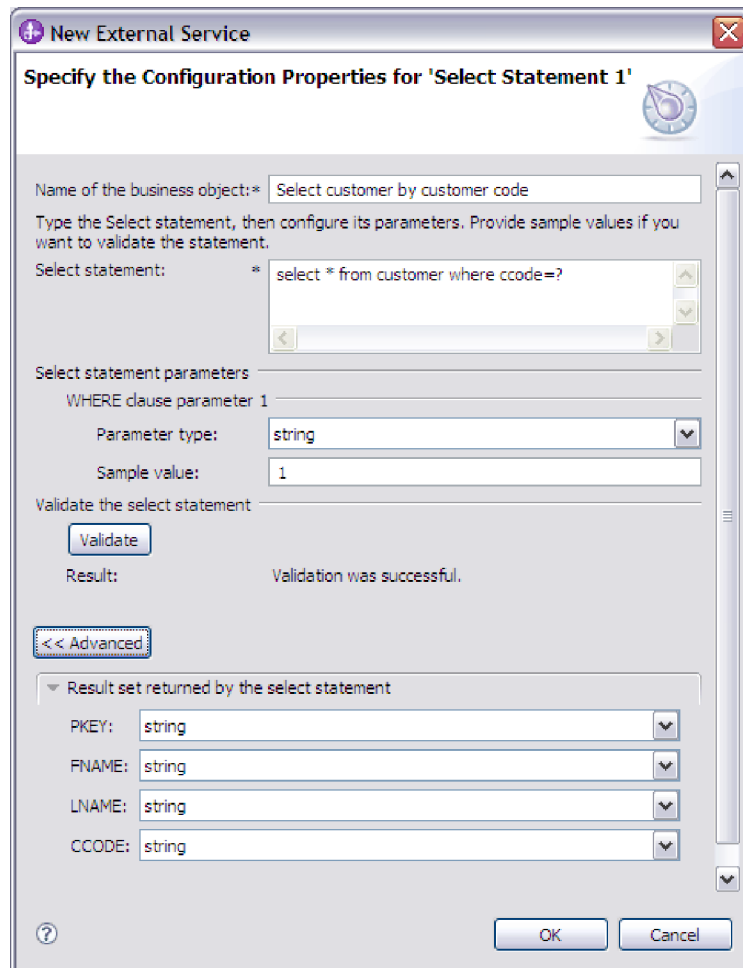
As you type each ?, the window expands to display an area where you define the WHERE clause for that parameter. The following figure shows the Specify the Configuration Properties for 'object' window for a query business object that has a single parameter.



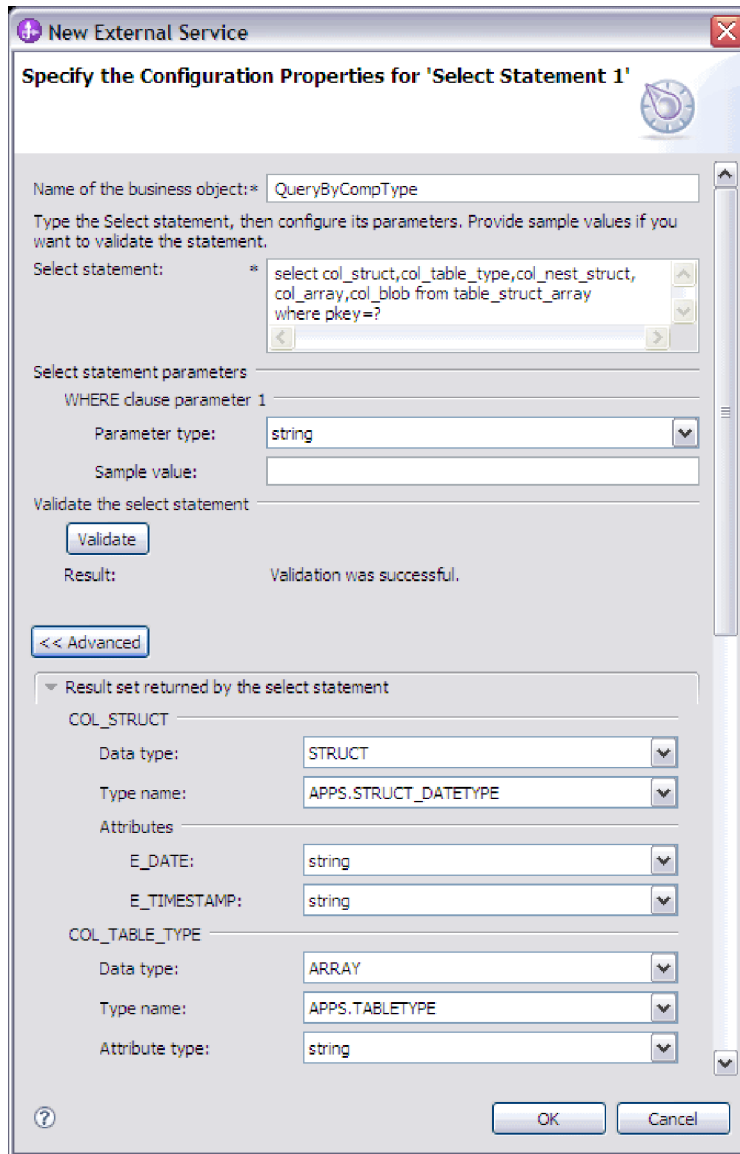
5. In the **WHERE clause parameter *n*** area, provide information about each parameter in the SELECT statement.
 - a. In the **Parameter type** field, select the data type of the parameter. For Oracle databases, the adapter does not support the complex types such as array, table, structure, or nested structure as parameters in the batch and query business objects.
 - b. In the **Sample value** field, type a sample value for the parameter. For example, for a parameter corresponding to a column containing a customer's last name, you might select string as the data type and provide a sample value of Smith.
6. Click **Validate**. The **Result** area displays the result of the validation. If the **Result** area displays Validation failed, there is a problem in the information you provided. Use the error message from the database server, which follows Validation failed, to correct the definition. Check the syntax of

the SELECT statement, the data type of the parameters, and the sample data. If the validation is successful, the **Advanced** button is displayed.

7. To specify the data type mapping for each column in the result set returned by the select statement:
 - a. Click **Advanced**.
 - b. Expand **Result set returned by the Select statement**. For each column in the result set, the default data type mapping is displayed.



For Oracle databases, if the query result contains any complex data type such as an array, structure, nested structure or table, the type name and the child attribute details are also automatically discovered and displayed. The figure below displays the type name and child attribute details in a query result for an Oracle table.



- c. Review the mapping and make changes if required.
8. Click **OK** to save the definition of the query business object.

Results

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

What to do next

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

Business object attributes

Business object attributes define the content of a business object and are built from the list of columns in the database object. Each attribute has a name, type, cardinality, and several other properties. The external service wizard sets the attribute name to the name of the column. The adapter adds the attribute cardinality, type, and application-specific information.

A business object is a container for the data specified in the attributes. The structure of the data in the database is defined by the business object, but data in the database is in the business object attributes.

Table 2 lists the properties of a business object attribute and describes their interpretation and settings.

Table 2. Attribute properties

Properties	Interpretation and settings
Cardinality	<p>An integer specifying the cardinality of a business object. Each business object attribute that represents a child or an array of child business objects has the value of single (1) or multiple (an unbounded integer) cardinality.</p> <p>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.</p>
Foreign Key	<p>When arrays of child business objects whose cardinality is n are retrieved, foreign keys are used in the WHERE clause of SELECT statements.</p> <p>The RetrieveAll operation overrides the use of keys and foreign keys.</p> <p>Note: The adapter does not support specifying an attribute that represents a child business object as a foreign key.</p>
Name	<p>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.</p>
MinOccurs MaxOccurs	<p>If the column is not a primary key and is not nullable, the MinOccurs and MaxOccurs attributes are required, and their values are set to at least 1.</p>
Primary Key	<p>Indicates whether this attribute is a primary key. At least, one simple attribute in each business object must be specified as the primary key.</p> <p>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.</p> <p>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.</p>

Table 2. Attribute properties (continued)

Properties	Interpretation and settings
Required	Specifies whether an attribute must contain a value. If this property is set to true for a container whose cardinality is single (1), then the adapter requires that the parent business object contain a child business object for this attribute. Business objects that are passed to the adapter for Create, Update, and Delete operations must also contain a child business object. Cardinality is single (1) for simple attributes and multiple (n) for container attributes. The adapter causes a Create operation to fail if a business object does not have a valid value or a default value for a required attribute. It also fails if no data is available upon retrieval from the database for this object.
Type	For simple attributes, this property specifies the type of the attribute, such as Integer, String, Date, Timestamp, Boolean, Double, or Float. The supported types for simple attributes and their mapping to the JDBC type of a database object are described in Table 3. For attributes that specify a child business object, this property specifies the name of the business object.

The type of each database object, returned as the JDBC metadata, maps to the business object attribute types as listed in Table 3. Only the JDBC types listed are supported by the adapter. Any columns with types that are not listed are not added to the business object. An informational message is produced explaining the problem, for example, The column named xxxx in the table named yyyy is not of a supported type and will not be added to the business object.

Table 3. JDBC metadata column type and business object attribute types

JDBC metadata column type	Business object attribute type
BIT	Boolean
CHAR LONGVARCHAR VARCHAR	String
NUMERIC	Decimal Int
INTEGER SMALLINT TINYINT	Int
BIGINT	Long Int
TIME	String Time
TIMESTAMP	String DateTime
DATE	String Date Datetime
DECIMAL	Decimal
DOUBLE FLOAT	Double
REAL	Float

Table 3. JDBC metadata column type and business object attribute types (continued)

JDBC metadata column type	Business object attribute type
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String
RAW	hexBinary
MONEY SMALLMONEY	Decimal
STRUCT or ARRAY	<p>The adapter processes these data types as child business objects of the table or query business objects.</p> <p>Note: The adapter supports complex types for Oracle table and query business objects only. If the table contains any complex data type such as an array, structure, nested structure or table, the type name and the sub attribute details are also automatically discovered and displayed.</p> <p>Note: The adapter treats an empty complex column as null irrespective of setting it to null or unset.</p>
XML	<p>String</p> <p>For Oracle, the following additional libraries are required to handle the XML data type in the runtime environment: xdb.jar xmlparserv2.jar</p> <p>In addition, if you are using a JNDI datasource to connect to the database, ensure that you add xdb.jar and xmlparserv2.jar to the class path when you create the datasource.</p> <p>Note: The XML type is supported in table business object only.</p> <p>Example of an XML content:</p> <pre><customer> <fname>John</fname> <lname>Smith</lname> </customer></pre> <p>For more information, see "Support for XML data type in Oracle" on page 67.</p>

Chapter 3. XML data type in table business objects

The adapter now supports the XML type in table business objects. If the table column in the database is of the XML type, the XML application-specific information is set to True. The XML metadata type maps to the String business object attribute type.

Adding external software dependencies

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

Before you begin

Run the external service wizard in WebSphere Integration Developer to perform this task.

About this task

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

Procedure

1. Obtain the JDBC driver-specific files or native libraries for your database software and operating system from your database administrator or from the database software Web site. The files that you need vary by database server. Ensure that the JDBC driver-specific files are compatible with JRE 1.6. The following table lists the JDBC driver files needed for common database software.

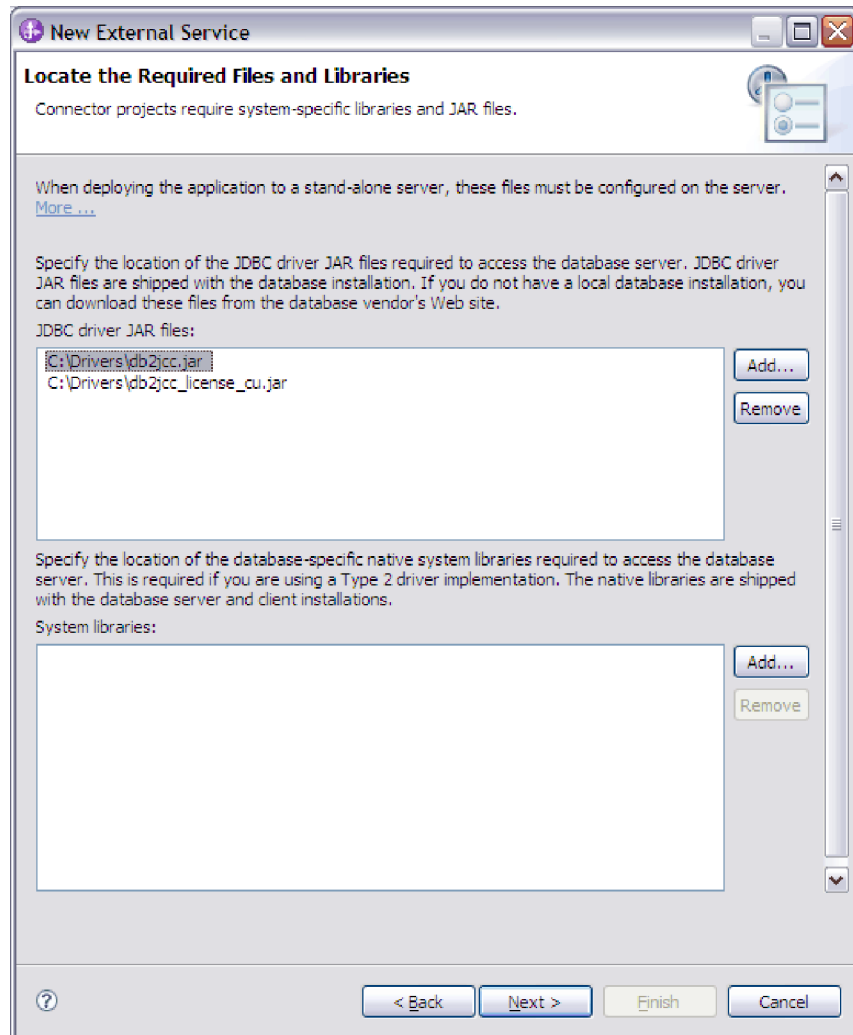
Table 4. JDBC driver files for common database software

Database software	Driver	JDBC driver files	Native System Libraries
IBM® DB2 Universal Database for Linux®, UNIX®, and Windows®	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	None
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	None
IBM DB2 for i5/OS®	IBM Toolbox for Java™ remote driver (Type 4)	jt400.jar db2jcc_license_cisuz.jar	None
	IBM Toolkit for Java native driver* (Type 2)	db2_classes.jar	None
IBM DB2 Universal Database for Linux, UNIX, and Windows	IBM DB2 Universal (Type 2)	db2java.zip	None

Table 4. JDBC driver files for common database software (continued)

Database software	Driver	JDBC driver files	Native System Libraries
Oracle	Thin driver	ojdbc6.jar In addition, the following additional libraries are required to handle the XML data type in the runtime environment: xdb.jar xmlparserv2.jar For more information, see "Support for XML data type in Oracle" on page 67.	None
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None
<p>* You can use the IBM Toolkit for Java native driver to connect to the database at adapter run time, but you cannot use it to connect while running the wizard. You must use either the IBM Toolbox for Java remote driver or the IBM DB2 Universal driver during the discovery process. However, you can configure the module to use the native driver at run time. Do this on the Specify the Service Generation and Deployment Properties window.</p>			

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



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

Results

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

What to do next

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

Business object attributes

Business object attributes define the content of a business object and are built from the list of columns in the database object. Each attribute has a name, type, cardinality, and several other properties. The external service wizard sets the attribute name to the name of the column. The adapter adds the attribute cardinality, type, and application-specific information.

A business object is a container for the data specified in the attributes. The structure of the data in the database is defined by the business object, but data in the database is in the business object attributes.

Table 5 lists the properties of a business object attribute and describes their interpretation and settings.

Table 5. Attribute properties

Properties	Interpretation and settings
Cardinality	<p>An integer specifying the cardinality of a business object. Each business object attribute that represents a child or an array of child business objects has the value of single (1) or multiple (an unbounded integer) cardinality.</p> <p>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.</p>
Foreign Key	<p>When arrays of child business objects whose cardinality is n are retrieved, foreign keys are used in the WHERE clause of SELECT statements.</p> <p>The RetrieveAll operation overrides the use of keys and foreign keys.</p> <p>Note: The adapter does not support specifying an attribute that represents a child business object as a foreign key.</p>
Name	<p>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.</p>
MinOccurs MaxOccurs	<p>If the column is not a primary key and is not nullable, the MinOccurs and MaxOccurs attributes are required, and their values are set to at least 1.</p>
Primary Key	<p>Indicates whether this attribute is a primary key. At least, one simple attribute in each business object must be specified as the primary key.</p> <p>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.</p> <p>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.</p>
Required	<p>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.</p>

Table 5. Attribute properties (continued)

Properties	Interpretation and settings
Type	<p>For simple attributes, this property specifies the type of the attribute, such as Integer, String, Date, Timestamp, Boolean, Double, or Float. The supported types for simple attributes and their mapping to the JDBC type of a database object are described in Table 6.</p> <p>For attributes that specify a child business object, this property specifies the name of the business object.</p>

The type of each database object, returned as the JDBC metadata, maps to the business object attribute types as listed in Table 6. Only the JDBC types listed are supported by the adapter. Any columns with types that are not listed are not added to the business object. An informational message is produced explaining the problem, for example, The column named xxxx in the table named yyyy is not of a supported type and will not be added to the business object.

Table 6. JDBC metadata column type and business object attribute types

JDBC metadata column type	Business object attribute type
BIT	Boolean
CHAR LONGVARCHAR VARCHAR	String
NUMERIC	Decimal Int
INTEGER SMALLINT TINYINT	Int
BIGINT	Long Int
TIME	String Time
TIMESTAMP	String DateTime
DATE	String Date Datetime
DECIMAL	Decimal
DOUBLE FLOAT	Double
REAL	Float
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String

Table 6. JDBC metadata column type and business object attribute types (continued)

JDBC metadata column type	Business object attribute type
RAW	hexBinary
MONEY SMALLMONEY	Decimal
STRUCT or ARRAY	<p>The adapter processes these data types as child business objects of the table or query business objects.</p> <p>Note: The adapter supports complex types for Oracle table and query business objects only. If the table contains any complex data type such as an array, structure, nested structure or table, the type name and the sub attribute details are also automatically discovered and displayed.</p> <p>Note: The adapter treats an empty complex column as null irrespective of setting it to null or unset.</p>
XML	<p>String</p> <p>For Oracle, the following additional libraries are required to handle the XML data type in the runtime environment: xdb.jar xmlparserv2.jar</p> <p>In addition, if you are using a JNDI datasource to connect to the database, ensure that you add xdb.jar and xmlparserv2.jar to the class path when you create the datasource.</p> <p>Note: The XML type is supported in table business object only.</p> <p>Example of an XML content: <customer> <fname>John</fname> <lname>Smith</lname> </customer></p> <p>For more information, see "Support for XML data type in Oracle" on page 67.</p>

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 7 on page 39.

Table 7. Application-specific information for simple attributes

Parameter	Type	Description	Default value
BLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the BLOB data type. While displaying BLOB data, the adapter displays the number of bytes as a hexadecimal value. The attribute type is hexBinary. If set to True, the column data type is BLOB.	None
ByteArray	Boolean	Specifies whether the column is a binary data type. If True, the adapter reads and writes binary data to the database and sends that data as a string to the application server. The adapter sets binary data on the business object. The attribute type is hexBinary.	False
ChildBOType	String	If the attribute is a complex data type, use this application-specific information to specify the actual type: <ul style="list-style-type: none"> • Struct • Array • ResultSet 	None
ChildBOTypeName	String	When the value of the ChildBOType application-specific information is either Struct or Array, this parameter represents the name of the user-defined type. This value is case-sensitive.	
CLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the CLOB data type. This value applies only to attributes of type String. If True, the column data type is CLOB. The CLOB attribute has a String type whose length is used to define the length of the CLOB.	None
ColumnName	String	The name of the database column corresponding to this attribute. This is the only required parameter.	None

Table 7. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
CopyAttribute	String	<p>A user-specified value that refers to another attribute name from within the same business object or parent business object.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>	None
DateType	String	<p>Specifies that the corresponding element is a date, time, or time stamp. Specify one of the following values:</p> <ul style="list-style-type: none"> • Date • Time • Timestamp <p>When setting the value of an attribute of the DateType type, use the following formats:</p> <ul style="list-style-type: none"> • For Date, use <i>yyyy-MM-dd</i> • For Time, use <i>hh:mm:ss</i> • For Timestamp, use <i>yyyy-MM-dd hh:mm:ss.fffffffff</i> <p>Note: The <code>java.sql.Timestamp.valueOf()</code> method converts the JDBC timestamp format to the Timestamp value and inserts this value into the database. The <code>java.sql.Timestamp.toString()</code> method converts the Timestamp value in the database to string. The adapter uses the Timestamp value present in the database. To learn more about the Timestamp method, see the Sun Web site at http://java.sun.com/j2se/1.5.0/docs/api/ and search for <i>Timestamp</i>.</p>	None
DateFormat	String	<p>Allows you to customize the format of the Date, Time and Timestamp data types. The adapter uses this parameter when the SQL Date, Time or Timestamp data type has to be converted to string and the other way around.</p>	None
DecimalScale	Int	<p>Specifies the scale of decimal data type. For example, $unscaledVal \times 10^{-scale}$</p>	None

Table 7. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
Dummy	Boolean	Indicates a dummy column. If True, the Dummy column value is not updated or inserted into the database. Use this application-specific information when you want to configure multiple ForeignKey values on one column.	None
FixedChar	Boolean	<p>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.</p> <p>This parameter must be updated manually in the XSD file in the business object. Open the business object by using an XML or Text editor to edit the XSD file and make the following two changes:</p> <ol style="list-style-type: none"> 1. Remove the type="string" added by default to the <element> tag for the object attribute. 2. Add a new <simpletype> section before the </element> as shown in this example: <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="10"/> </xsd:restriction> </xsd:simpleType></pre> <p>Save the object definition, and ensure that no validation errors occur in the XSD file after it has been updated.</p> <p>See the section "Example of a FixedChar parameter in the business object XSD file" following this table.</p>	false
ForeignKey	String	<p>The value of this property depends on whether the parent and child relationship is stored in the parent business object or in the child.</p> <p>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 (<i>Child_BO_name/Child_Property_Name</i>).</p> <p>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.</p> <p>If an attribute is not a foreign key, do not include this parameter in the application-specific information.</p>	None
OrderBy	String	<p>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.</p> <p>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.</p>	None

Table 7. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
PrimaryKey	Boolean	If the column associated with this attribute is a primary key in the corresponding table in the database, the PrimaryKey parameter is set to True.	None
SPParameterType	String	Specifies the type of stored procedure Possible values are: <ul style="list-style-type: none"> • IP (input only) • OP (output only) • IO (input and output) • RS (result set) 	None
UniqueIdentifier (UID)	String	<p>The adapter uses this parameter to generate the unique ID for the business object. It supports the generation of sequences and identity columns (identity columns are known as <i>serial columns</i> in Informix). DB2 supports both sequences and identity columns.</p> <p>Identity columns provide a way for the database to automatically generate a unique numeric value for each row in a table.</p> <p>Identity columns can be defined for DB2 and Microsoft SQL Server, and serial columns can be defined for Informix.</p> <p>The format of this parameter is as follows:</p> <p>UID=AUTO <i>Sequence_Name</i></p> <p>If you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, you must manually set the UID (Unique Identifier) attribute to AUTO, for example, <UID>AUTO</UID>.</p> <p>Note: The requirement to manually set the UID (Unique Identifier) attribute to AUTO is specific to identity columns in DB2 and Microsoft SQL Server. The requirement does not apply to serial columns in Informix. For Informix, the UID attribute for the serial column is generated automatically and will be either <UID>SERIAL</UID> or <UID>SERIAL8</UID>.</p> <p>Like identity columns, sequences are also used to automatically generate numeric values. Consult your database documentation for details on how the database uses sequences and identity columns.</p> <p>For a sequence, set the UID attribute to the name of the sequence. Sequences can be defined for DB2 and Oracle databases.</p> <p>If the attribute does not require a unique ID, do not include this parameter in the application-specific information.</p>	None
XML	Boolean	If the table column in the database is of the XML type, the XML parameter is set to True.	None

The format of attribute application-specific information is shown in the following example section of an XSD file:

Example section of an XSD file

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

Example of FixedChar parameter in the business object XSD file

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

Application-specific information for attributes that refer to child business object

Two application-specific information parameters are used for attributes that refer to child business objects (complex, as opposed to simple, attributes). When you set this application-specific information, specify the parameters listed in Table 8 on page 44.

Table 8. Application-specific information for attributes of type child business object

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

Example of ownership in the business object XSD file

```
<element minOccurs="0" name="addressObj" type="bons0:OutboundRtasserAddress"
maxOccurs="unbounded">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:Ownership>true</jdbcasi:Ownership>
      </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>
```

```
<element minOccurs="0" name="custInfoObj" type="bons1:OutboundRtasserCustInfo"
maxOccurs="1">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:Ownership>>false</jdbcasi:Ownership>
      </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>
```

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

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

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

```

        <pasi:Ownership>false</pasi:Ownership>
    </pasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>

```

Application-specific information for operations

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

If you choose to add stored procedures or stored functions to the business objects, set the operation application-specific information (ASI) as specified in Table 9.

Table 9. 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: <ul style="list-style-type: none"> The string RS. This value indicates that the procedure returns a result set, which is used to create the multiple-cardinality container corresponding to this business object. The name of a business object attribute. This value indicates that procedure returns the value that is to be assigned to that particular attribute in the business object at run time. If the attribute is another child business object, the adapter returns an error.
StoredProcedure	Yes	Set to the stored procedure name.
StoredProcedureType	Yes	You select from a list of types. For information about valid stored procedure types, see Stored procedure type.
Type	Yes	Set to the type of the stored procedure parameter. Possible values are: <ul style="list-style-type: none"> IP (input only) OP (output only) IO (input and output) RS (result set)

Solutions to common problems

Some of the problems you might encounter while running WebSphere Adapter for JDBC with your database are described along with solutions and workaround. These problems and solutions are in addition to those documented as technotes on the software support Web site.

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

Cannot add object to selection

Problem

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

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

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

Cause

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

Solution

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

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

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

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

Problem

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

- Oracle 9i: ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: An operation on the database failed with a SQL exception for the following reason: No more data to read from socket.
- Oracle 10g: ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: An operation on the database failed with a SQL exception for the following reason: ORA-01460: unimplemented or unreasonable conversion requested

Cause

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

Solution

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

Some generated business objects have no attributes for Oracle database objects

Problem

Some business objects that are generated from an Oracle database object do not have attributes for the table columns.

Cause

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

- 2281705. DATABASEMETADA.GETCOLUMNS does not return underlying table if there is a synonym
- 2696213. JDBC GETPROCEDURECOLUMNS does not return columns for the synonym of a procedure

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

Solution

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

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

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

Using the adapter to connect to IBM DB2 for z/OS® with a JDBC (Type 2 or Type 4) universal driver

Problem and Cause

DB2 for z/OS supports querying stored procedure metadata by using positional index by default and not using column name, which the Adapter for JDBC uses. The solution provides the steps for using the Adapter for JDBC with DB2 on the z/OS platform.

Solution

To connect to DB2 for z/OS using the Adapter for JDBC, ensure that the following connection requirements are met:

- The physical representation of the universal JDBC driver is the db2jcc.jar file. Ensure that the path to this file is set in the class path.
- Database URL: To determine whether you are using the Type 2 or Type 4 driver, review the form of the connection:

Type 2: jdbc:db2:database

(For example: jdbc:db2:MyDB, where MyDB is the database name)

Type 4: jdbc:db2://server:port/database

(For example: jdbc:db2://9.182.15.129:50000/MyDB,, where MyDB is the database name)

- Driver class: com.ibm.db2.jcc.DB2Driver.

The driver class for both Type 2 and Type 4 drivers is the same.

- Set the path of the db2jcc_license_cisuz.jar file in the class path.

The license JAR file is the same for both Type 2 and Type 4 drivers. Access to DB2 for z/OS and DB2 for i5/OS servers requires a valid DB2 Connect™ license. DB2 clients do not provide direct connectivity to zSeries® and iSeries® servers without a DB2 Connect license.

For more information about DB2 Connect licensing and usage, refer to the following pages:

<http://www-128.ibm.com/developerworks/db2/library/techarticle/0303zikopoulos1/0303zikopoulos1.html>

<http://www-128.ibm.com/developerworks/db2/library/techarticle/0301zikopoulos/0301zikopoulos.html>

There might be issues with importing metadata for stored procedures using the wizard. To use stored procedures and import metadata from DB2 using the Adapter for JDBC, DB2 needs to be reconfigured as described in the following steps. Follow these steps in addition to the steps provided above:

- Apply the following APARs on DB2: PQ62695, PQ55393, PQ56616, PQ54605, PQ46183, and PQ62139.
- If you want to use stored procedures with the adapter, follow the steps below, which are part of the fix for PQ62695. This fix introduces stored procedures that provide the ability to generate a result set that corresponds to the schema metadata APIs documented in the JDBC and ODBC specification.

These procedures are used by the JDBC and ODBC drivers provided in the DB2 Universal Driver. Follow these steps to enable support for stored procedures:

1. Apply the APAR.
2. Check the value of the DESCSTAT variable in the ZPARM assembly job DSNTIJUZ. If the value of the DESCSTAT variable is NO, change it to YES.

Note: The default for DESCSTAT is NO on V7 but was changed to YES on V8.

3. Reassemble and reinitialize the ZPARM module.

4. Run the JCL job named DSNTIJMS. You can find this member in the db2prefix.SDSNSAMP data set.
5. Restart DB2.

Using XA transactions for outbound support with a remote DB2 database

Using XA Transactions with the Adapter for JDBC using the Universal Driver

The following versions of software and configuration properties are needed to use XA transactions with the Adapter for JDBC and the Universal driver to connect to a remote DB2 database:

- DB2 version: 8.2 or later
- JDBC Driver: UDB driver Type 4 and Type 2
- XA data source name: com.ibm.db2.jcc.DB2XADataSource
- XA Database name: This is the remote database alias configured on the local DB2 client.
- Database URL: jdbc:db2://hostname:port/databasename
- JDBC driver class: com.ibm.db2.jcc.DB2Driver

Using XA transactions for Outbound support with a remote DB2 database

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

Using XA Transactions on a remote DB2 database

Adding a remote DB2 database

1. Run the db2admin (*DB2_InstallPath*\SQLLIB\BIN) command on the DB2 server.
2. Open the DB2 Configuration Assistant.
3. Go to **View** → **Advanced View**.

Complete the following steps, in order:

1. **Add the remote system**
 - a. Select **Systems** tab.
 - b. From the menu, select **Selected** → **Add System**.
 - c. In the **System name** field, specify the physical machine, server system, or workstation where the target database is located. The system name on the server system is defined by the DB2SYSTEM DAS configuration parameter. This is the value that you should use.
 - d. In the **Host name** field, type the host name or Internet Protocol (IP) address where the target database resides.
 - e. In the **Node name** field, specify a local nickname for the remote node where the database is located. The node name you selected must not already exist in the node directory or the admin node directory.
 - f. Select the operating system and click **OK**.
2. **Add an instance node**
 - a. Select the **Instance Nodes** tab.
 - b. From the menu, select **Selected** → **Add Instance Node**.

- c. In the **System name** field, specify the physical machine, server system, or workstation where the target database is located. Select the system added in the Adding a remote system task.
 - d. In the **Instance name** field, type the name of the instance (DB2, and so on) where the target database is located.
 - e. In the **Instance node name** field, specify a unique nickname for the cataloged system (node) where the database is located. The node name you selected must not already exist in the node directory or the admin node directory.
 - f. Select the operating system and type the host name. Use the same host name as in step 4 of the task: Adding the remote system.
 - g. Enter the port number on which the remote DB2 instance is running.
 - h. Click **OK**.
3. **Add a database**
- a. Select the **Databases** tab.
 - b. From the menu, select **Selected** → **Add Database**.
 - c. In the **Instance node** field, select the instance created in the task: Adding an instance node. Specify the name of the database that you are adding in the **Database name** field.
 - d. In the **Alias** field, specify a local nickname that can be used by applications running on your workstation. If nothing is entered, the alias will be the same as the database name. The alias name should be unique.

Note: This alias name value should be entered in the XADatabaseName property for the adapter.

4. **Test the database connection**
- a. Select the **Databases** tab.
 - b. Choose the database added in the task: Adding a database.
 - c. From the menu, select **Selected** → **Test Connection**.
 - d. Select the **CLI** check box, type the user ID and password and click **Test Connection**. This should return a successful connection.

A closer look at the transaction (XID) column in the event table

If the adapter is configured for assured once delivery, use the status column with the XID column to determine whether the event has been processed:

- If the XID column contains 0, the event has not yet been picked up for processing.
- If the XID column contains a transaction ID (that is, it does not contain 0), then the adapter has started to process the event but has not completed processing. You might see this combination when the adapter or application server crashes while the event is being processed. The transaction manager will either COMMIT or ROLLBACK these transactions during recovery.

Handling unexpected results from a query SQL statement

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

Configuring SQL Server 2000 to support XA transaction

To configure SQL SERVER 2000 to support XA transaction:

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

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

Cause

This is a limitation with SQL Server 2000 JDBC driver.

Solution

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

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

Multiple JDBC adapter export components in the same SCA module

When multiple JDBC adapter export components exist in the same SCA module, these exports do not pick up and operate on the same event record from same event table. If you define multiple exports to access the same event record, potential problems like database deadlock error might occur. You have to configure different `EventTypeFilter` (Event types to process) condition on each export for multiple JDBC Adapter export components picking up records from the same event table so that the risk of one event record being picked up by several exports is eliminated.

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

Cause

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

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

Here is the detail information about the database metadata in the `jtoopen_6_4_source\com\ibm\as400\access\doc-files\JDBCProperties.html` from the JTOpen download Web site : <http://sourceforge.net/projects/jt400/files/JTOpen-full/6.4/>

Solution

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

Changing the schema without regenerating the artifacts

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

To make changes to the import or export file:

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

Below is the code snippet for `JDBCInboundInterface.export`

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

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

To make changes to the *.xsd files:

1. Open the *.xsd file in the text editor. For example, `JcajdbcCustomer.xsd`

Below is the code snippet for `JcajdbcCustomer.xsd`:

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

```

<jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>RS</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>

```

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

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

Problem

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

Solution

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

Stored procedure name BIDI support for MS SQL Server

Problem

For MS SQL server, if the stored procedure name is in a language written from right-to-left, the business object configuration window for the stored procedure is not displayed. The log file displays the exception "java.lang.RuntimeException: com.microsoft.sqlserver.jdbc.SQLServerException: Conversion failed when converting the nvarchar value '??' to data type int". If the EMD wizard is completed ignoring this exception, the corresponding stored procedure business object is not generated.

Solution

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

Schema name BIDI support for Oracle database

Problem

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

Solution

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

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

Problem

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

Cause

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

Solution

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

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

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

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

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

WebSphere Adapter for JDBC and MySQL database support

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

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

Problem

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

Cause

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

Solution

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

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

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

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

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

Problem

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

Solution

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

Incorrect binary data is returned when querying DB2/AS400

Problem

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

Cause

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

Solution

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

Error when using stored procedure with return value

Problem

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

Cause

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

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

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi"
asiNSURI="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="PocuserGetpolicycount">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnValue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>returnValue</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
```

```

</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="returnvalue" type="int" minOccurs="1" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@RETURN_VALUE</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="ipolicynum" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

Solution

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

```

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi"
asiNSURI="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="PocuserGetpolicycount">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnvalue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>

```

```

</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="ipolicynum" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

Class loader violation occurs when starting the external service wizard

Problem

It is not possible to use the external service wizard after using a connection to the database in the Data perspective. At the end of the second panel of the wizard, the following exception is generated:

```

com.ibm.adapter.framework.api.ImportException
Reason: class loading constraint violated (class:
oracle/jdbc/driver/OracleConnection
method: getWrapper()Loracle/jdbc/OracleConnection;) at pc:0

```

The error occurs in both of the following situations:

- When you establish a connection to the database through the external service wizard, an error occurs when you attempt to connect to the database from the Data perspective.
- When you establish a connection to the database through the Data perspective, the error occurs when you attempt to connect to the database through the external service wizard.

Cause

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

Solution

The only solution is to restart WebSphere Integration Developer when this error occurs.

Closed connection error occurs when using XA with Oracle 10g

Problem

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

Cause

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

Solution

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

Error while starting a transaction on Oracle

Problem

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

Solution

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

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

This activity is likely to be performed by your Oracle database administrator, because you must be logged into Oracle as SYSOPER or SYSDBA in order to have the necessary permissions for these scripts to work.

The `initxa.sql` script configures the database for XA. Once it runs successfully, your database is configured for XA. The script might run successfully the first time you try. It might not run successfully if the database memory space is too small.

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

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

Table 10 on page 60 shows two parameters that typically need to be increased. Your particular database configuration might require adjustment of different parameters.

Table 10. Typical parameter sizes

Parameter Name	Minimum Value
java_pool_size	12000000
shared_pool_size	24000000

ResourceException during outbound processing

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

- SQLException exception

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

For example:

```
javax.resource.ResourceException: [ibm][db2][jcc][t4][2013][11249] Connection authorization failure occurred. Reason: User ID or Password invalid.
```

- ConnectException exception

If the ConnectException includes the text connection refused or could not establish connection to the server, then the database server might not be operational or there might be a network problem that prevents a connection.

For example:

```
javax.resource.ResourceException: [ibm][db2][jcc][t4][2043][11550] Exception java.net.ConnectException: Error opening socket to server /9.26.237.55 on port 50,000 with message: Connection refused: connect.
```

ResourceException during inbound processing

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

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

Problem

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

Solution

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

WebSphere Adapter for JDBC throws RecordNotFoundException

Problem

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

Cause

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

Problem

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

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

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

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

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

See "WebSphere Adapter for JDBC throws RecordNotFoundException" on page 60.

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

Problem

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

Cause

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

Solution

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

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

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

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

Solution

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

See also, [Selecting and configuring stored procedures and stored functions.](#)

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

See [Selecting and configuring stored procedures and stored functions.](#)

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

Problem

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

Solution

For those columns whose SQL type is DATE, TIME, and TIMESTAMP, JDBC Enterprise Metadata Discovery sets the corresponding attribute type to string, and an attribute application-specific information named DataType is added. Its value is

set to either Date, Time, or TimeStamp. To generate the business object XSDs with the proper attribute type and application-specific information, run the JDBC Enterprise Metadata Discovery again.

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

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

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

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

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

BPEL cannot catch specific business fault

See BPEL cannot catch specific Business Fault

Outbound fails due to an exception in global transaction

Problem

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

Cause

From version 6.2 onwards, WebSphere Adapter for JDBC supports business objects for exceptions including IntegrityConstraintViolationException, MatchesExceededLimitException, MissingDataException, MultipleMatchingRecordsException, ObjectNotFoundException, RecordNotFoundException, and UniqueConstraintViolatedException. When any of these exceptions are thrown, the EIS binding encapsulates the exceptions as corresponding fault business objects like IntegrityConstraintFault, MatchesExceededLimitFault, MissingDataFault, MultipleMatchingRecordsFault, ObjectNotFoundException, RecordNotFoundException, and UniqueConstraintFault.

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

Solution

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

```

<methodBinding
  inDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2admin
ncustom erbg.Db2adminCustomerBGDataBinding"
  method="createDb2adminCustomerBG"
  outDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2adm
incusto merbg.Db2adminCustomerBGDataBinding">
  <faultBinding fault="INTEGRITY_CONSTRAINT_VIOLATION"
  faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
BindingI mpl"/>
  <faultBinding fault="MISSING_DATA"
  faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
BindingI mpl"/>
  <faultBinding fault="OBJECT_NOTFOUND_EXCEPTION"
  faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
BindingI mpl"/>
  <faultBinding fault="UNIQUECONSTRAINT_VIOLATION"
  faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
BindingI mpl"/>
  <interaction>
    <properties>
      <functionName>Create</functionName>
    </properties>
  </interaction>
</methodBinding>

```

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

```

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

```

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

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

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

1. In the Business Integration perspective of WebSphere Integration Developer, select **File > New > External Service**.
2. Select IBM WebSphere Adapter for JDBC and create an Outbound or Inbound service. Ensure that you add the JDBC drivers to the adapter connector project in order to connect to various databases. If you want the JDBC adapter to connect to different databases, you can add all the JDBC drivers to the JDBC adapter connector project at once.
3. Select the exiting connector project and create another Outbound or Inbound service. Use the same module that you created earlier for the artifacts.
4. After you create the Inbound or Outbound services, check whether there is only JDBC adapter connector project listed in the **J2EE** area of Dependencies page for the selected module.

For more information, see [Sharing a JDBC adapter with one or more modules in an ear file](#).

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

Problem

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

Solution

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

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

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

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

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

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

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

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

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

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

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

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

Problem

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

Cause

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

Solution

Here are the migration steps.

1. Create table "WBIA_JDBC_EVENTSTORE" with the connector_ID.

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

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

2. Create and run the below SQL statements.

user-defined query:

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

user-defined update:

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

user-defined delete:

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

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

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

JDBC adapter cannot find method for native method during inbound process

Problem

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

Cause

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

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

Solution

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

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

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

Problem

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

Cause

DB2 JCC driver version is 4.7.85.

Solution

Use DB2 JCC driver 3.50.152.

Oracle Date type mapped to dateTime instead of date

Problem

In Oracle, the Date data type is mapped to dateTime instead of date by default for some fields.

Cause

This is an issue with the Oracle JDBC driver. In Oracle database, the Date type is similar to the Timestamp type defined in JDBC specification. The Date type also contains the time information. The adapter depends on the data type returned from the driver to map the JDBC type to the SDO type. If the driver returns the Date JDBC type, the adapter maps it to the date SDO type. If the driver returns the Timestamp JDBC type, then adapter map it to the dateTime SDO type.

Solution

You can manually map it to the required type.

Support for XML data type in Oracle

Problem1

When xmlparserv2.jar is added to the application library, an exception is generated in the administrative console when you are trying to deploy the application.

Cause

When xmlparserv2.jar is added to the application library, it registers as an implementation of XML processing interfaces, DocumentBuilderFactory, SAXParserFactory and TransformerFactory. When xmlparserv2.jar is registered as the TransformerFactory implementation, due to compatibility issue the following exception is generated in the application deployment phase:

```
java.lang.IllegalArgumentException
  at oracle.xml.jaxp.JXTransformer.setOutputProperty(JXTransformer.java:793)
  at org.eclipse.xsd.util.DefaultJAXPConfiguration.createTransformer
    (DefaultJAXPConfiguration.java:63)
  at org.eclipse.xsd.util.XSDResourceImpl.doSerialize(XSDResourceImpl.java:153)
  at org.eclipse.xsd.util.XSDResourceImpl.serialize(XSDResourceImpl.java:136)
  at org.eclipse.xsd.ecore.XSDEcoreBuilder.setAnnotations(XSDEcoreBuilder.java:3087)
  at com.ibm.ws.bo.BOModelBuilder.access$201(BOModelBuilder.java:103)
  at com.ibm.ws.bo.BOModelBuilder$1.run(BOModelBuilder.java:1778)
  at java.security.AccessController.doPrivileged(AccessController.java:202)
  at com.ibm.ws.bo.BOModelBuilder.setAnnotations(BOModelBuilder.java:1774)
  at org.eclipse.xsd.ecore.XSDEcoreBuilder.getEPackage(XSDEcoreBuilder.java:161)
  at com.ibm.ws.bo.BOModelBuilder.getEStructuralFeature(BOModelBuilder.java:983)
  at org.eclipse.xsd.ecore.XSDEcoreBuilder.generate(XSDEcoreBuilder.java:2709)
  at com.ibm.ws.bo.BOModelBuilder.generate(BOModelBuilder.java:340)
  at com.ibm.ws.bo.BOModelBuilder.generate(BOModelBuilder.java:650)
  at com.ibm.ws.bo.BOModelBuilder.build(BOModelBuilder.java:309)
  at com.ibm.ws.bo.BOModelHolder.loadModels(BOModelHolder.java:591)
  at com.ibm.ws.bo.BOModelHolder.loadAllModels(BOModelHolder.java:626)
  at com.ibm.ws.bo.BOModelHolder.loadNamespace(BOModelHolder.java:545)
  at com.ibm.ws.bo.BOEPackageRegistry.loadEPackage(BOEPackageRegistry.java:218)
  at com.ibm.ws.bo.BOEPackageRegistry.getEPackage(BOEPackageRegistry.java:295)
```

Solution

Force the classloader to load the compatible implementation by renaming jaxp.properties.sample to jaxp.properties under <WPS_HOME>/java/jre/lib/. In addition, in the jaxp.properties file, uncomment the properties javax.xml.transform.TransformerFactory, javax.xml.parsers.SAXParserFactory, and javax.xml.parsers.DocumentBuilderFactory. Restart the server.

Note: For Oracle, if you are using a JDNI datasource to connect to the database, ensure that you add xdb.jar and xmlparserv2.jar to the class path when you are create the datasource.

Problem2

In Oracle (driver version 11.1.0.7.0), for a table with xml data type, the Retrieve and RetrieveAll operations returns incorrect result.

Cause

In Oracle driver version 11.1.0.7.0, the content of the XML type attribute in a complex type is not read correctly. The driver always returns null.

Solution

Use driver version 11.2.0.1.0.

Chapter 4. Customize format for date, time, and timestamp data types

You can customize the format of the Date, Time and Timestamp data types using the DateFormat parameter. The adapter uses this parameter when the SQL Date, Time or Timestamp data type has to be converted to string and the other way around.

Business objects

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

How the adapter uses business objects

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

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

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

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

- Batch SQL business objects, which represent a series of user-defined insert, update, and delete statements.
- Query business objects, which represent a user-defined SQL query to run against the database.
- Wrapper business objects, which allow you group unrelated table and view objects into a single business object, and multiple stored procedures into a single business object.

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

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

ExistsResult	
status	boolean
recordcount	int

How data is represented in business objects

For table or view business objects

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

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

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

A table business object always has a primary key, even if the corresponding database table does not have a primary key. The adapter uses the column specified in the primary key attribute when it retrieves table business objects. If you have defined foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables. For example, consider tables CUSTOMER and ADDRESS, where CUSTOMER is the parent table and ADDRESS is the child table. If you have defined a foreign key reference from ADDRESS to CUSTOMER in the database, the adapter automatically discovers the parent-child relationship displays the foreign key reference in the Specify the Configuration Properties for 'object' window. If the foreign key reference is from CUSTOMER to ADDRESS, the adapter automatically selects the **Single cardinality** check box and displays the foreign key reference. If there are multiple foreign key references defined for a table, the adapter generates only one foreign key relationship.

The adapter supports tables that have composite, or multiple primary keys. If a database table has one or more primary keys, the wizard sets the primary key property for those columns in the table business object. If the database table does not have a primary key, the external service wizard prompts you for primary key information when you discover and configure that business object. If there is a

composite primary key reference, for instance CUSTOMER (pkey1, pkey2) > ADDRESS (fkey1, fkey2), the adapter associates ADDRESS (fkey1) with CUSTOMER (pkey1) and ADDRESS (fkey2) with CUSTOMER (pkey2). Specify a column that contains unique data, such as a sequence or identity column. Identity columns (known as *serial columns* in Informix; JDBC adapter supports both serial and serial8) provide a way for the database to automatically generate a unique numeric value for each row in a table. A table can have a single column that is defined with the identity attribute. Examples of an identity column include order number, employee number, stock number, and incident number. Identity columns can be defined for tables in DB2, Informix and Microsoft SQL Server only.

Note: When you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object for that table does not include the Unique Identifier attribute of the identity column. In this case, you need to edit the generated business object by adding the attribute to the application-specific information manually. You can do this through the assembly editor in WebSphere Integration Developer. You do not need to add the attribute for the Unique Identifier manually if you ran the discovery process against a table in an Informix database. For Informix, the generated business object includes the Unique Identifier attribute of the serial column.

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.fffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

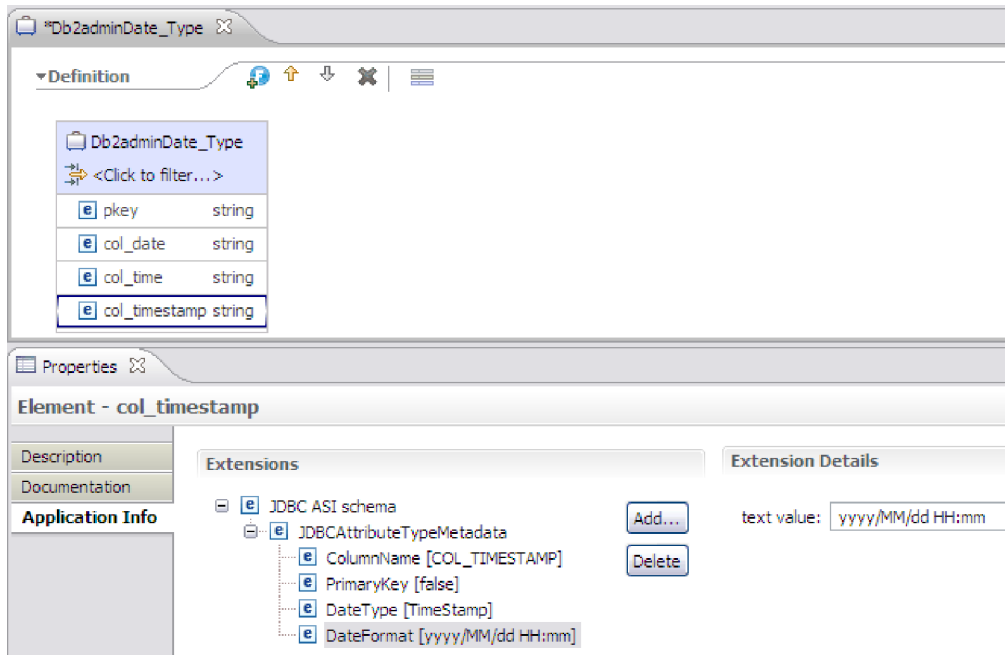


Figure 11. The DateFormat application-specific information with the customized format

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

Figure 12 shows a table business object in the business object editor. The business object has an attribute for each of the columns in the database table. Because the table has no child business objects, all of the attributes are simple attributes.

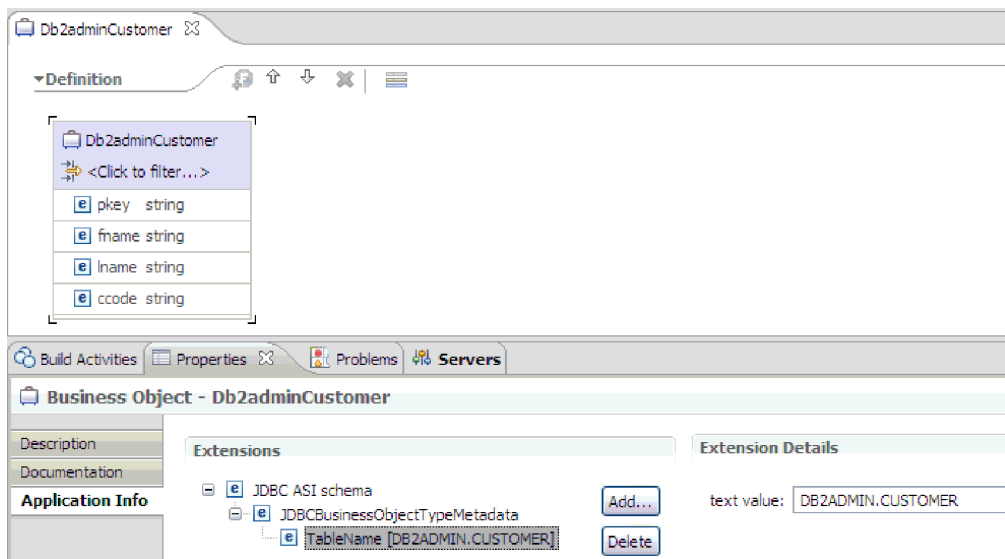


Figure 12. A table business object with no child.

Figure 13 on page 73 shows a table business object that has one child table business object. The business object has simple attributes for each of the columns

in the database table, plus a complex attribute pointing to a child business object.

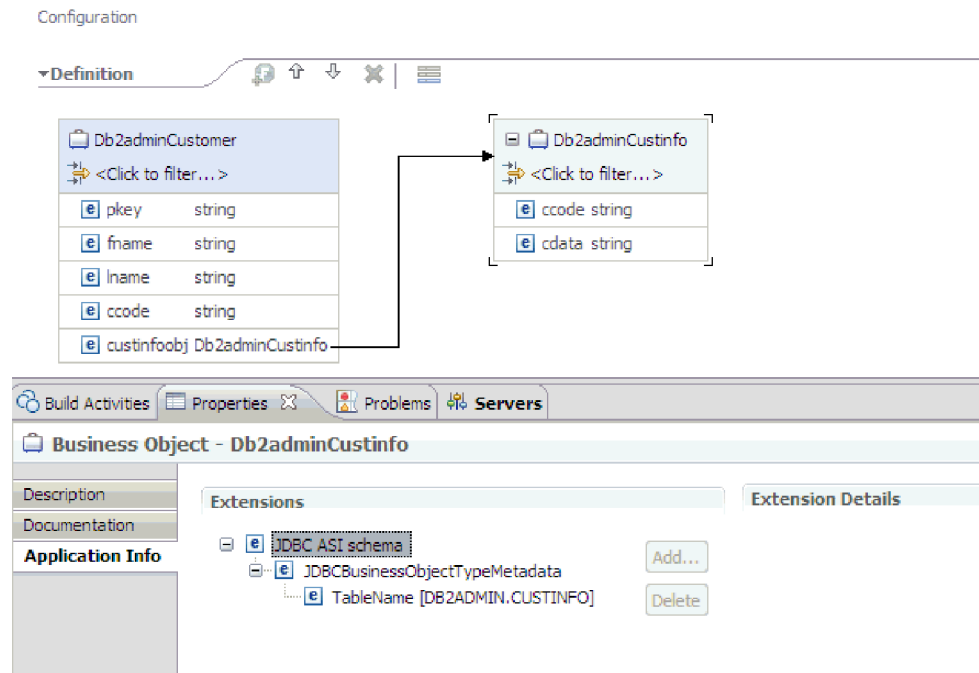


Figure 13. A table business object with one child business object.

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

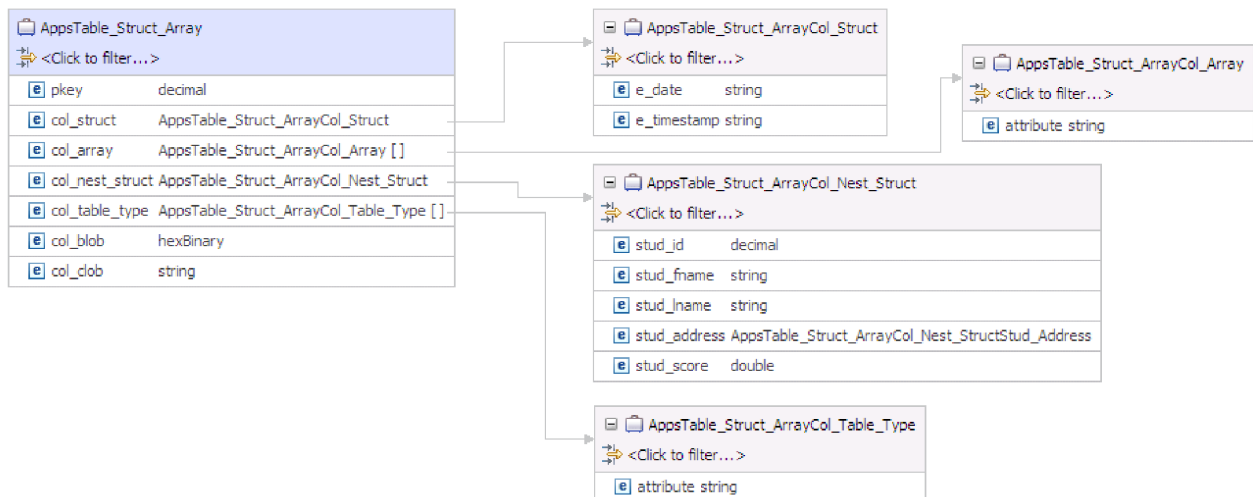


Figure 14. An Oracle table business object having user-defined or complex types as columns

For stored procedure and stored function business objects

In a business object for a stored procedure or stored function, all the input and output parameters for the stored procedure or stored function have corresponding

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

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.ffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

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

Example of ScottStrtValues business object

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
```



```

<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>IP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
<jdbcasi:ChildBOType>STRUCT</jdbcasi:ChildBOType>
<jdbcasi:ChildBOTypeName>STRUCT1</jdbcasi:ChildBOTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

Example of ScottStrtValuesStrt business object

```

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asi:SURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>

```

```

</annotation>
<sequence minOccurs="1" maxOccurs="1">
  <element name="name" type="string" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
  <element name="title" type="string" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
  <element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPPParameterType></jdbcasi:SPPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
</sequence>
</complexType>
</schema>

```

For query business objects

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

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

```

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

```

The question mark (?) indicates an input parameter for the query. A query can have multiple parameters, each indicated in the SELECT statement by a question mark. Table 11 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 11. 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

Table 11. Attributes of a query business object (continued)

Business object attribute	Description
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

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the dd/MM/yy format and timestamp in the yyyy/MM/dd HH:mm format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.ffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

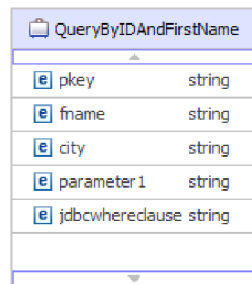


Figure 15. The attributes of a query business object

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

statement.

For Oracle databases, the adapter supports complex data types such as array, table,

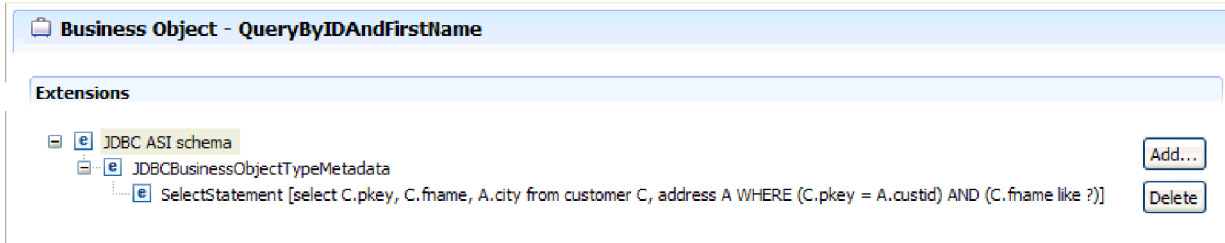


Figure 16. The SELECT statement is saved in the business object application-specific information

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

For batch SQL business objects

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

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

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

Each question mark (?) indicates a parameter for the statement. Each statement in a batch SQL business object can have multiple parameters, each indicated in the statement by a question mark. A batch SQL business object can have multiple statements, each with its own set of parameters. Figure 17 shows the format of the business object for the batch SQL business object with an INSERT and a DELETE statement, each of which has one or more parameters.

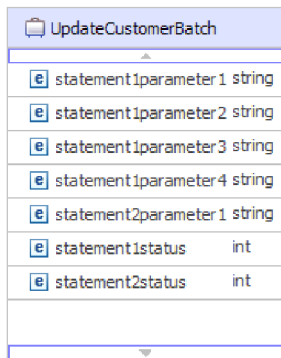


Figure 17. A batch SQL business object with two SQL statements

The business object has an attribute for each parameter in each statement, including statement1parameter1, statement2parameter1, and so on. It also has an attribute for the status of each statement, such as statement1status, statement2status, and so on. The statements themselves are stored as application-specific information about the business object, as shown in Figure 18 on page 79

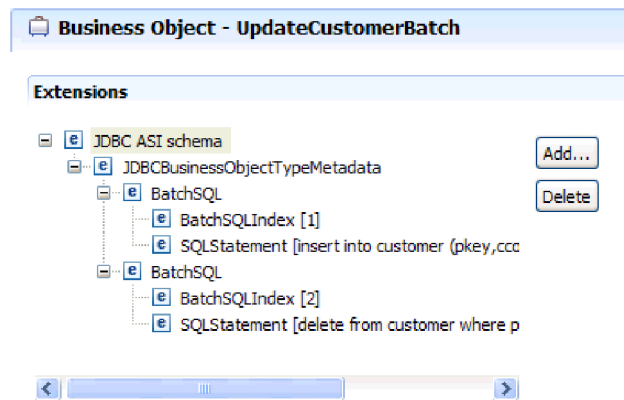


Figure 18. The application-specific information of a batch SQL business object

For wrapper business objects

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

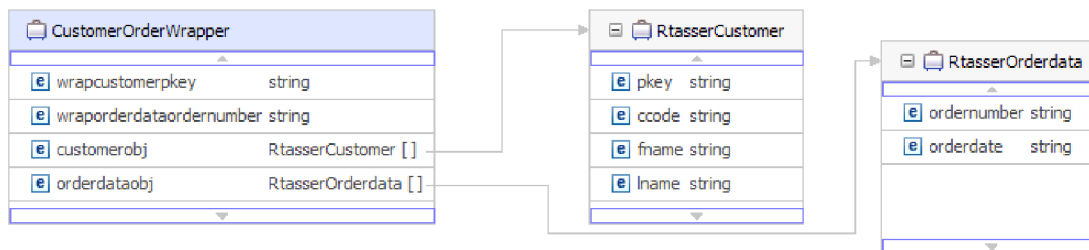


Figure 19. A wrapper business object that contains two table business objects

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

Business graphs

You can optionally choose, during adapter configuration, to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use in version 6.0.2 to specify additional information about the operation to be performed. In version 7.0, business graphs are required only in these situations:

- If you need to use the outbound ApplyChanges operation
- When adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 7.0

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

How business objects are created

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

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

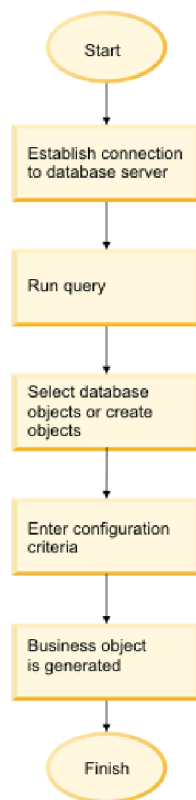


Figure 20. How business objects are created

In some instances, the wizard cannot completely configure a parent-child relationship. For these relationships, you use the business objects editor, launched from WebSphere Integration Developer, to modify or complete the definition of a business object hierarchy that was created by the wizard. For more information, see the instructions for using the business object editor to modify business objects in the WebSphere Integration Developer information center at the following link: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp>

Discovering database objects for outbound processing

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

Before you begin

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

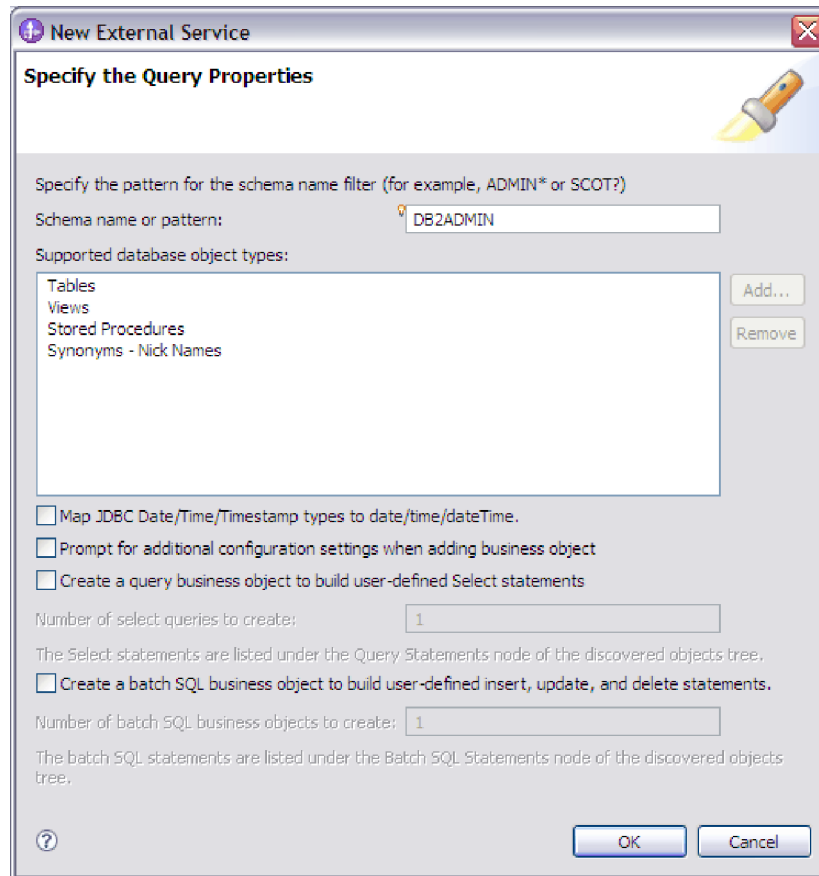
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas
- Which tables, view, synonyms or nicknames, stored procedures or stored functions you need to access
- How many query and batch SQL business objects you need to create, including parameter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

About this task

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

Procedure

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



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

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

Note: In version 6.1.x with fix pack 2, this window also allowed you to specify the number of wrapper business objects you want to create. In version 6.2.x, the wizard prompts for wrapper information at a later time.

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, stored procedures and stored functions, and synonyms or nicknames) in the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If you

need to access only specific types of database objects, omitting the ones you do not need can speed up the discovery process.

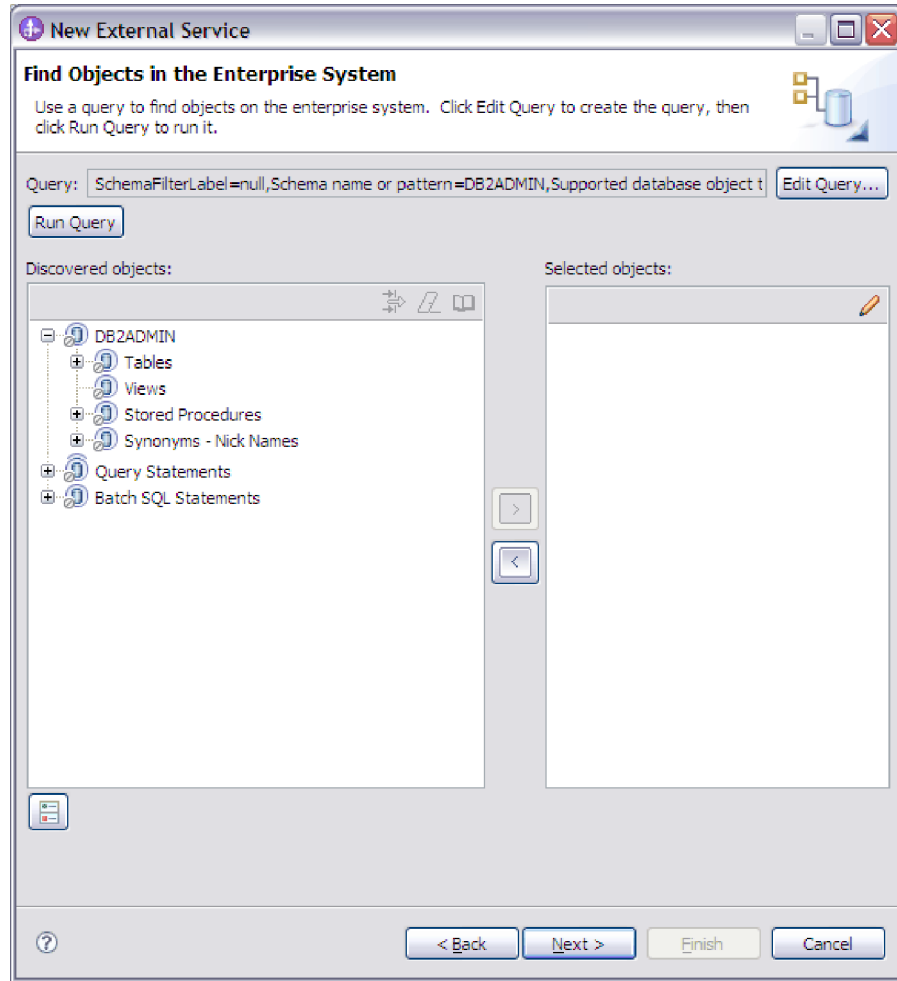
4. The table, stored procedure, and stored function objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.

Note: The default data type mapping differs based on the different JDBC driver versions, for example, when using the Oracle JDBC driver, the Date data type is mapped to dateTime data type instead of mapping it to date. In such cases, the appropriate data type must be manually selected in the Specify the Configuration Properties for 'object' window.

5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when you add a database object to the list of business objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple parent-child hierarchy of business objects. If you need a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is launched from WebSphere Integration Developer. In addition, if a foreign key reference is defined in the database, the adapter automatically discovers and displays the parent-child relationship between the tables.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor. In addition, if you have not defined a foreign key reference in the database, the adapter will not generate the parent-child relationship automatically.

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



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

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

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

Results

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

What to do next

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

Discovering database objects for inbound processing

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

Before you begin

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

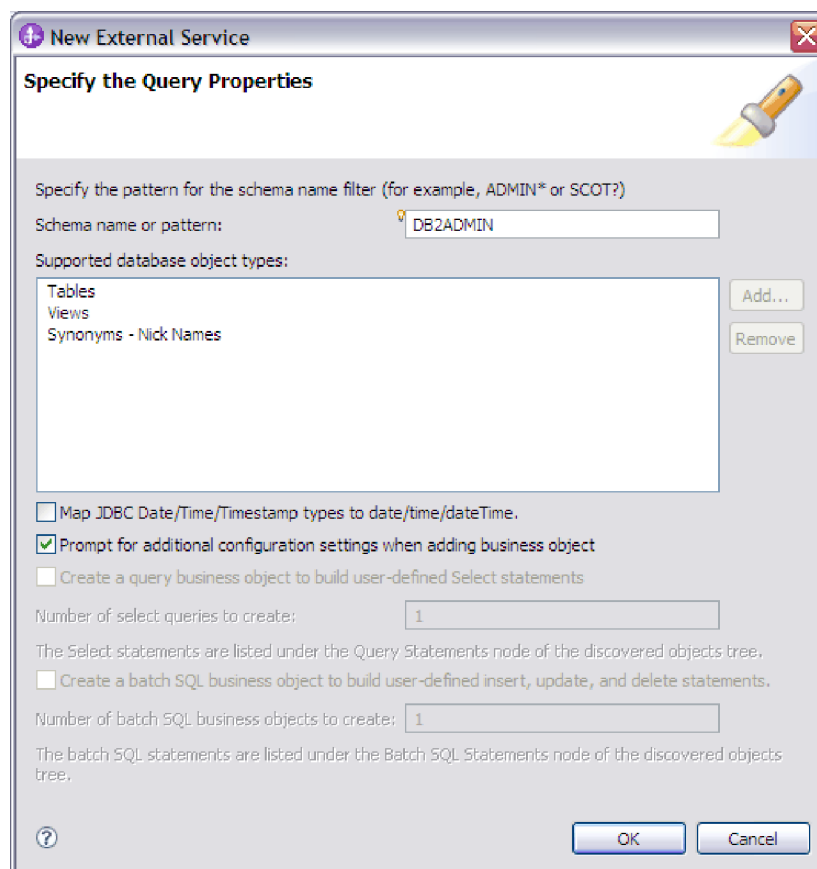
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas

About this task

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

Procedure

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



Note: The options **Create a query business object to build user-defined select statements** and **Create a batch SQL business object to build user-defined insert, update, and delete statements** are available only for outbound processing.

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

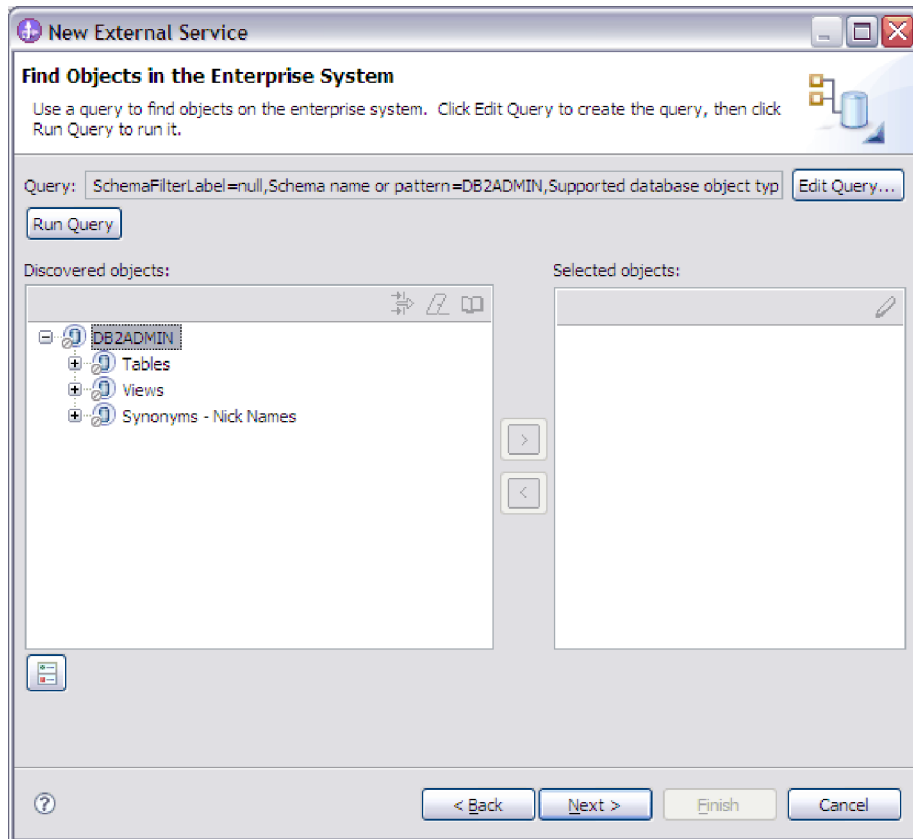
- Reduce the search time by searching a subset of database schemas
 - Omit one or more types of database objects from the search
 - Make the wizard prompt you for application-specific information that cannot be automatically determined based on information in the database
 - Map the JDBC data types Date, Time and Timestamp to date, time and dateTime
2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
 3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, and synonyms or nicknames) in the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If your database contains object types that you do not need to access, omitting them can speed up the discovery process.
 4. Table objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.

Note: The default data type mapping differs based on the different JDBC driver versions, for example, when using the Oracle JDBC driver, the Date data type is mapped to dateTime data type instead of mapping it to date. In such cases, the appropriate data type must be manually selected in the Specify the Configuration Properties for 'object' window.

5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when you add a database object to the list of business objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple parent-child hierarchy of business objects. If you need a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is launched from WebSphere Integration Developer. In addition, if a foreign key reference is defined in the database, the adapter automatically discovers and displays the parent-child relationship between the tables.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor. In addition, if you have not defined a foreign key reference in the database, the adapter will not generate the parent-child relationship automatically.

6. Click **OK** to save your changes to the query.
7. In the Find Objects in the Enterprise System window, click **Run Query** to use the query to discover database objects. The result of running a typical query is shown in the following figure.



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

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

Results

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

What to do next

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

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 12.

Table 12. Application-specific information for simple attributes

Parameter	Type	Description	Default value
BLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the BLOB data type. While displaying BLOB data, the adapter displays the number of bytes as a hexadecimal value. The attribute type is hexBinary. If set to True, the column data type is BLOB.	None
ByteArray	Boolean	Specifies whether the column is a binary data type. If True, the adapter reads and writes binary data to the database and sends that data as a string to the application server. The adapter sets binary data on the business object. The attribute type is hexBinary.	False
ChildBOType	String	If the attribute is a complex data type, use this application-specific information to specify the actual type: <ul style="list-style-type: none"> • Struct • Array • ResultSet 	None
ChildBOTypeName	String	When the value of the ChildBOType application-specific information is either Struct or Array, this parameter represents the name of the user-defined type. This value is case-sensitive.	
CLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the CLOB data type. This value applies only to attributes of type String. If True, the column data type is CLOB. The CLOB attribute has a String type whose length is used to define the length of the CLOB.	None
ColumnName	String	The name of the database column corresponding to this attribute. This is the only required parameter.	None

Table 12. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
CopyAttribute	String	<p>A user-specified value that refers to another attribute name from within the same business object or parent business object.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>	None
DateType	String	<p>Specifies that the corresponding element is a date, time, or time stamp. Specify one of the following values:</p> <ul style="list-style-type: none"> • Date • Time • Timestamp <p>When setting the value of an attribute of the DateType type, use the following formats:</p> <ul style="list-style-type: none"> • For Date, use <i>yyyy-MM-dd</i> • For Time, use <i>hh:mm:ss</i> • For Timestamp, use <i>yyyy-MM-dd hh:mm:ss.fffffffff</i> <p>Note: The <code>java.sql.Timestamp.valueOf()</code> method converts the JDBC timestamp format to the Timestamp value and inserts this value into the database. The <code>java.sql.Timestamp.toString()</code> method converts the Timestamp value in the database to string. The adapter uses the Timestamp value present in the database. To learn more about the Timestamp method, see the Sun Web site at http://java.sun.com/j2se/1.5.0/docs/api/ and search for <i>Timestamp</i>.</p>	None
DateFormat	String	Allows you to customize the format of the Date, Time and Timestamp data types. The adapter uses this parameter when the SQL Date, Time or Timestamp data type has to be converted to string and the other way around.	None
DecimalScale	Int	Specifies the scale of decimal data type. For example, $unscaledVal \times 10^{-scale}$	None

Table 12. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
Dummy	Boolean	Indicates a dummy column. If True, the Dummy column value is not updated or inserted into the database. Use this application-specific information when you want to configure multiple ForeignKey values on one column.	None
FixedChar	Boolean	<p>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.</p> <p>This parameter must be updated manually in the XSD file in the business object. Open the business object by using an XML or Text editor to edit the XSD file and make the following two changes:</p> <ol style="list-style-type: none"> 1. Remove the type="string" added by default to the <element> tag for the object attribute. 2. Add a new <simpletype> section before the </element> as shown in this example: <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="10"/> </xsd:restriction> </xsd:simpleType></pre> <p>Save the object definition, and ensure that no validation errors occur in the XSD file after it has been updated.</p> <p>See the section "Example of a FixedChar parameter in the business object XSD file" following this table.</p>	false
ForeignKey	String	<p>The value of this property depends on whether the parent and child relationship is stored in the parent business object or in the child.</p> <p>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 (<i>Child_BO_name/Child_Property_Name</i>).</p> <p>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.</p> <p>If an attribute is not a foreign key, do not include this parameter in the application-specific information.</p>	None
OrderBy	String	<p>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.</p> <p>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.</p>	None

Table 12. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
PrimaryKey	Boolean	If the column associated with this attribute is a primary key in the corresponding table in the database, the PrimaryKey parameter is set to True.	None
SPParameterType	String	Specifies the type of stored procedure Possible values are: <ul style="list-style-type: none"> • IP (input only) • OP (output only) • IO (input and output) • RS (result set) 	None
UniqueIdentifier (UID)	String	<p>The adapter uses this parameter to generate the unique ID for the business object. It supports the generation of sequences and identity columns (identity columns are known as <i>serial columns</i> in Informix). DB2 supports both sequences and identity columns.</p> <p>Identity columns provide a way for the database to automatically generate a unique numeric value for each row in a table.</p> <p>Identity columns can be defined for DB2 and Microsoft SQL Server, and serial columns can be defined for Informix.</p> <p>The format of this parameter is as follows:</p> <p>UID=AUTO <i>Sequence_Name</i></p> <p>If you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, you must manually set the UID (Unique Identifier) attribute to AUTO, for example, <UID>AUTO</UID>.</p> <p>Note: The requirement to manually set the UID (Unique Identifier) attribute to AUTO is specific to identity columns in DB2 and Microsoft SQL Server. The requirement does not apply to serial columns in Informix. For Informix, the UID attribute for the serial column is generated automatically and will be either <UID>SERIAL</UID> or <UID>SERIAL8</UID>.</p> <p>Like identity columns, sequences are also used to automatically generate numeric values. Consult your database documentation for details on how the database uses sequences and identity columns.</p> <p>For a sequence, set the UID attribute to the name of the sequence. Sequences can be defined for DB2 and Oracle databases.</p> <p>If the attribute does not require a unique ID, do not include this parameter in the application-specific information.</p>	None
XML	Boolean	If the table column in the database is of the XML type, the XML parameter is set to True.	None

The format of attribute application-specific information is shown in the following example section of an XSD file:

Example section of an XSD file

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

Example of FixedChar parameter in the business object XSD file

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

Application-specific information for attributes that refer to child business object

Two application-specific information parameters are used for attributes that refer to child business objects (complex, as opposed to simple, attributes). When you set this application-specific information, specify the parameters listed in Table 13 on page 93.

Table 13. Application-specific information for attributes of type child business object

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

Example of ownership in the business object XSD file

```
<element minOccurs="0" name="addressObj" type="bons0:OutboundRtasserAddress"
maxOccurs="unbounded">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:Ownership>true</jdbcasi:Ownership>
      </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>
```

```
<element minOccurs="0" name="custInfoObj" type="bons1:OutboundRtasserCustInfo"
maxOccurs="1">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:Ownership>>false</jdbcasi:Ownership>
      </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>
```

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

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

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

```

        <pasi:Ownership>false</pasi:Ownership>
    </pasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>

```

Application-specific information for operations

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

If you choose to add stored procedures or stored functions to the business objects, set the operation application-specific information (ASI) as specified in Table 14.

Table 14. 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: <ul style="list-style-type: none"> • The string RS. This value indicates that the procedure returns a result set, which is used to create the multiple-cardinality container corresponding to this business object. • The name of a business object attribute. This value indicates that procedure returns the value that is to be assigned to that particular attribute in the business object at run time. If the attribute is another child business object, the adapter returns an error.
StoredProcedure	Yes	Set to the stored procedure name.
StoredProcedureType	Yes	You select from a list of types. For information about valid stored procedure types, see Stored procedure type.
Type	Yes	Set to the type of the stored procedure parameter. Possible values are: <ul style="list-style-type: none"> • IP (input only) • OP (output only) • IO (input and output) • RS (result set)

Chapter 5. Automatic discovery of parent-child relationship in tables

If you have defined a foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables when you select and configure the business object.

Business objects

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

How the adapter uses business objects

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

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

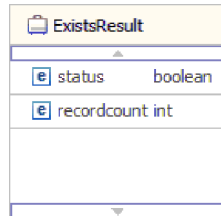
- Tables and views
- Synonyms and nicknames
- Stored procedures and stored functions

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

- Batch SQL business objects, which represent a series of user-defined insert, update, and delete statements.
- Query business objects, which represent a user-defined SQL query to run against the database.
- Wrapper business objects, which allow you group unrelated table and view objects into a single business object, and multiple stored procedures into a single business object.

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

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



How data is represented in business objects

For table or view business objects

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

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

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

A table business object always has a primary key, even if the corresponding database table does not have a primary key. The adapter uses the column specified in the primary key attribute when it retrieves table business objects. If you have defined foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables. For example, consider tables CUSTOMER and ADDRESS, where CUSTOMER is the parent table and ADDRESS is the child table. If you have defined a foreign key reference from ADDRESS to CUSTOMER in the database, the adapter automatically discovers the parent-child relationship displays the foreign key reference in the Specify the Configuration Properties for 'object' window. If the foreign key reference is from CUSTOMER to ADDRESS, the adapter automatically selects the **Single cardinality** check box and displays the foreign key reference. If there are multiple foreign key references defined for a table, the adapter generates only one foreign key relationship.

The adapter supports tables that have composite, or multiple primary keys. If a database table has one or more primary keys, the wizard sets the primary key property for those columns in the table business object. If the database table does not have a primary key, the external service wizard prompts you for primary key information when you discover and configure that business object. If there is a

composite primary key reference, for instance CUSTOMER (pkey1, pkey2) > ADDRESS (fkey1, fkey2), the adapter associates ADDRESS (fkey1) with CUSTOMER (pkey1) and ADDRESS (fkey2) with CUSTOMER (pkey2). Specify a column that contains unique data, such as a sequence or identity column. Identity columns (known as *serial columns* in Informix; JDBC adapter supports both serial and serial8) provide a way for the database to automatically generate a unique numeric value for each row in a table. A table can have a single column that is defined with the identity attribute. Examples of an identity column include order number, employee number, stock number, and incident number. Identity columns can be defined for tables in DB2, Informix and Microsoft SQL Server only.

Note: When you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object for that table does not include the Unique Identifier attribute of the identity column. In this case, you need to edit the generated business object by adding the attribute to the application-specific information manually. You can do this through the assembly editor in WebSphere Integration Developer. You do not need to add the attribute for the Unique Identifier manually if you ran the discovery process against a table in an Informix database. For Informix, the generated business object includes the Unique Identifier attribute of the serial column.

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the dd/MM/yy format and timestamp in the yyyy/MM/dd HH:mm format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.fffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

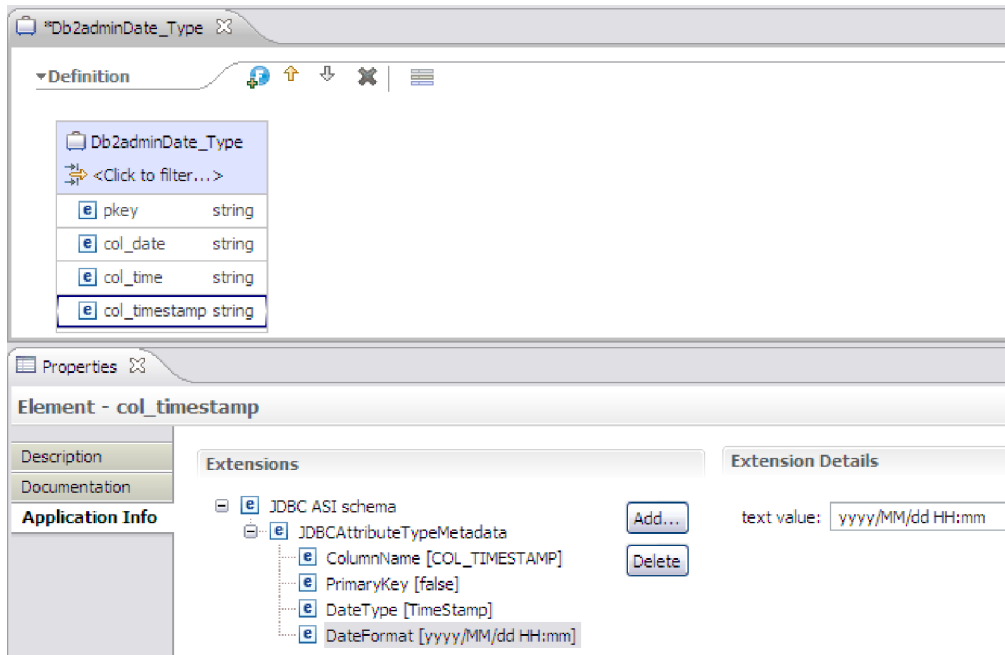


Figure 21. The DateFormat application-specific information with the customized format

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

Figure 22 shows a table business object in the business object editor. The business object has an attribute for each of the columns in the database table. Because the table has no child business objects, all of the attributes are simple attributes.

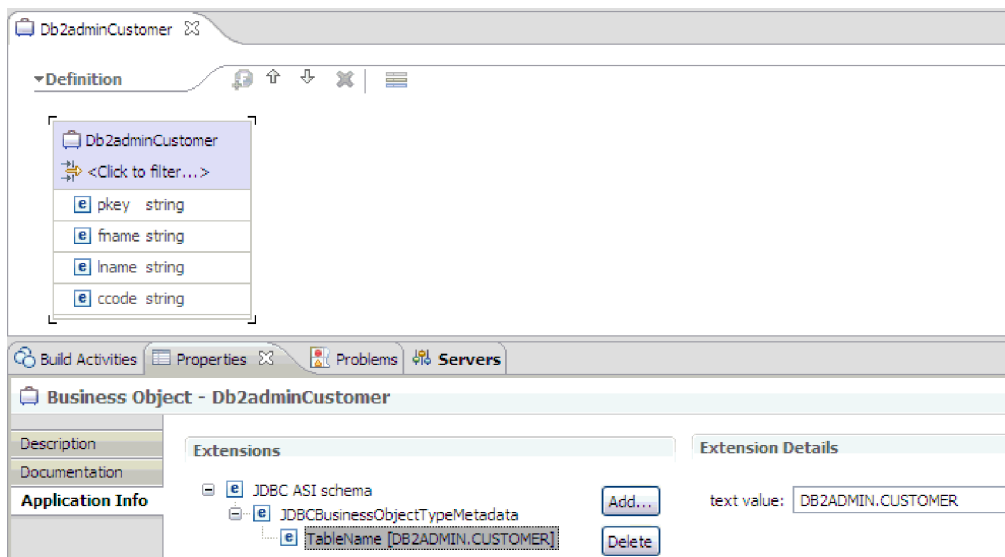


Figure 22. A table business object with no child.

Figure 23 on page 99 shows a table business object that has one child table business object. The business object has simple attributes for each of the columns

in the database table, plus a complex attribute pointing to a child business object.

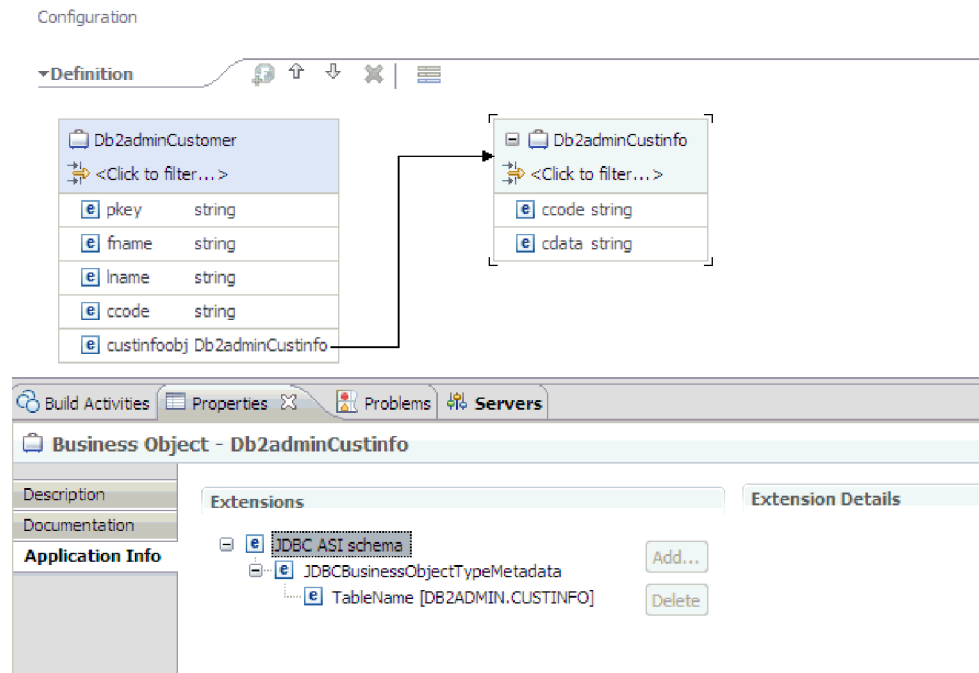


Figure 23. A table business object with one child business object.

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

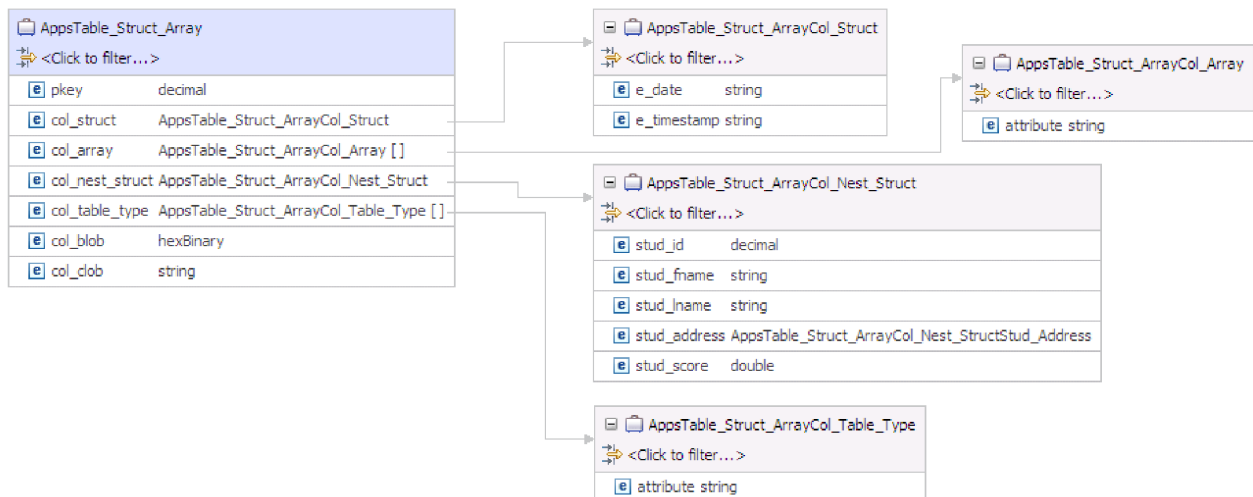


Figure 24. An Oracle table business object having user-defined or complex types as columns

For stored procedure and stored function business objects

In a business object for a stored procedure or stored function, all the input and output parameters for the stored procedure or stored function have corresponding

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

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the `dd/MM/yy` format and timestamp in the `yyyy/MM/dd HH:mm` format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.ffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

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

Example of ScottStrtValues business object

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
```

```

<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>IP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPPParameterType>OP</jdbcasi:SPPParameterType>
<jdbcasi:ChildBOType>STRUCT</jdbcasi:ChildBOType>
<jdbcasi:ChildBOTypeName>STRUCT1</jdbcasi:ChildBOTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>

```

Example of ScottStrtValuesStrt business object

```

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

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asi:SURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>

```

```

</annotation>
<sequence minOccurs="1" maxOccurs="1">
  <element name="name" type="string" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPParameterType></jdbcasi:SPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
  <element name="title" type="string" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPParameterType></jdbcasi:SPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
  <element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
    <annotation>
      <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
        <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
          "http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
          <jdbcasi:SPParameterType></jdbcasi:SPParameterType>
        </jdbcasi:JDBCAttributeTypeMetadata>
      </appinfo>
    </annotation>
  </element>
</sequence>
</complexType>
</schema>

```

For query business objects

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

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

```

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

```

The question mark (?) indicates an input parameter for the query. A query can have multiple parameters, each indicated in the SELECT statement by a question mark. Table 15 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 15. 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

Table 15. Attributes of a query business object (continued)

Business object attribute	Description
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

If the business object contains the Date, Time or Timestamp data type, the format of these types can be customized in the DateFormat application-specific information. The date format must follow the patterns defined in `java.text.SimpleDateFormat`. When an SQL Date, Time or Timestamp has to be converted to string and the other way around, and if you have customized the format for these types in the DateFormat application-specific information, the adapter uses this customized format. For example, you can specify the date in the dd/MM/yy format and timestamp in the yyyy/MM/dd HH:mm format. If the DateFormat application-specific information is not specified, the adapter uses the default format. The default format for the Date type is "yyyy-MM-dd", Timestamp type is "yyyy-mm-dd hh:mm:ss.ffffff", and Time type is "HH:mm:ss".

Note: The format for Timestamp type is defined in the JDBC specification and it does not follow the SimpleDateFormat pattern.

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

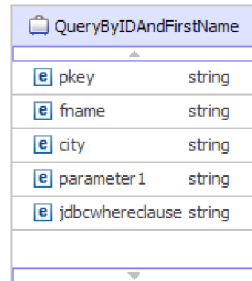


Figure 25. The attributes of a query business object

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

statement.

For Oracle databases, the adapter supports complex data types such as array, table,

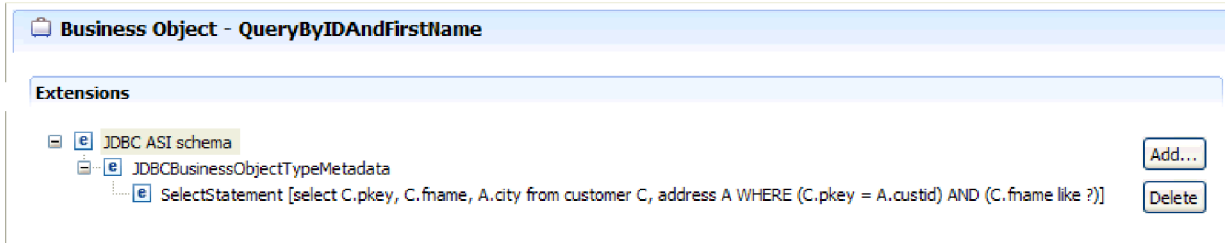


Figure 26. The SELECT statement is saved in the business object application-specific information

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

For batch SQL business objects

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

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

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

Each question mark (?) indicates a parameter for the statement. Each statement in a batch SQL business object can have multiple parameters, each indicated in the statement by a question mark. A batch SQL business object can have multiple statements, each with its own set of parameters. Figure 27 shows the format of the business object for the batch SQL business object with an INSERT and a DELETE statement, each of which has one or more parameters.

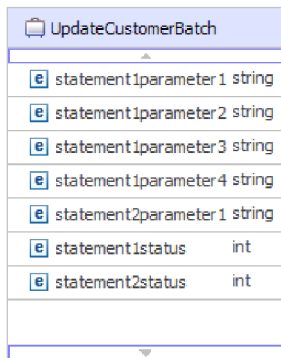


Figure 27. A batch SQL business object with two SQL statements

The business object has an attribute for each parameter in each statement, including statement1parameter1, statement2parameter1, and so on. It also has an attribute for the status of each statement, such as statement1status, statement2status, and so on. The statements themselves are stored as application-specific information about the business object, as shown in Figure 28 on page 105

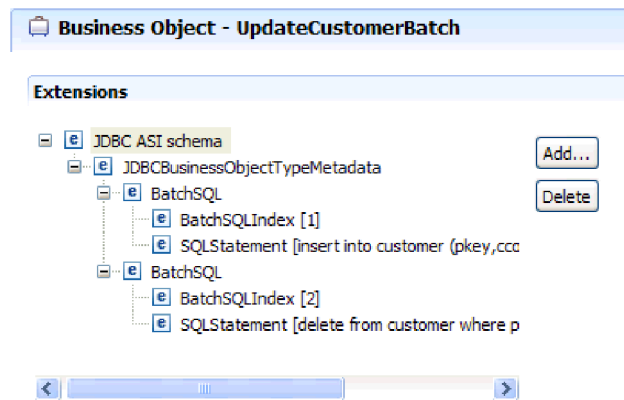


Figure 28. The application-specific information of a batch SQL business object

For wrapper business objects

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

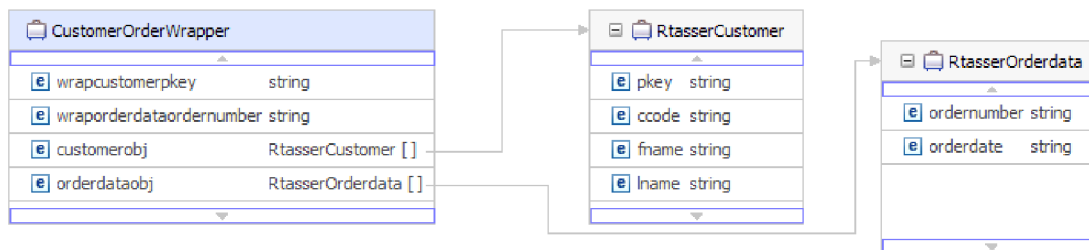


Figure 29. A wrapper business object that contains two table business objects

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

Business graphs

You can optionally choose, during adapter configuration, to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use in version 6.0.2 to specify additional information about the operation to be performed. In version 7.0, business graphs are required only in these situations:

- If you need to use the outbound ApplyChanges operation
- When adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 7.0

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

How business objects are created

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

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

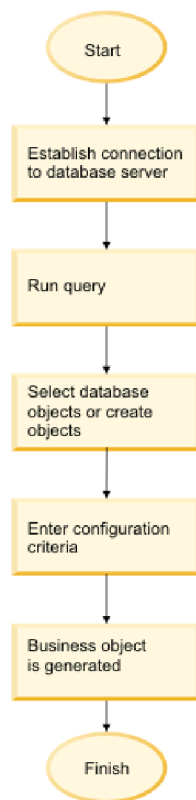


Figure 30. How business objects are created

In some instances, the wizard cannot completely configure a parent-child relationship. For these relationships, you use the business objects editor, launched from WebSphere Integration Developer, to modify or complete the definition of a business object hierarchy that was created by the wizard. For more information, see the instructions for using the business object editor to modify business objects in the WebSphere Integration Developer information center at the following link: <http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp>

Discovering database objects for outbound processing

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

Before you begin

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

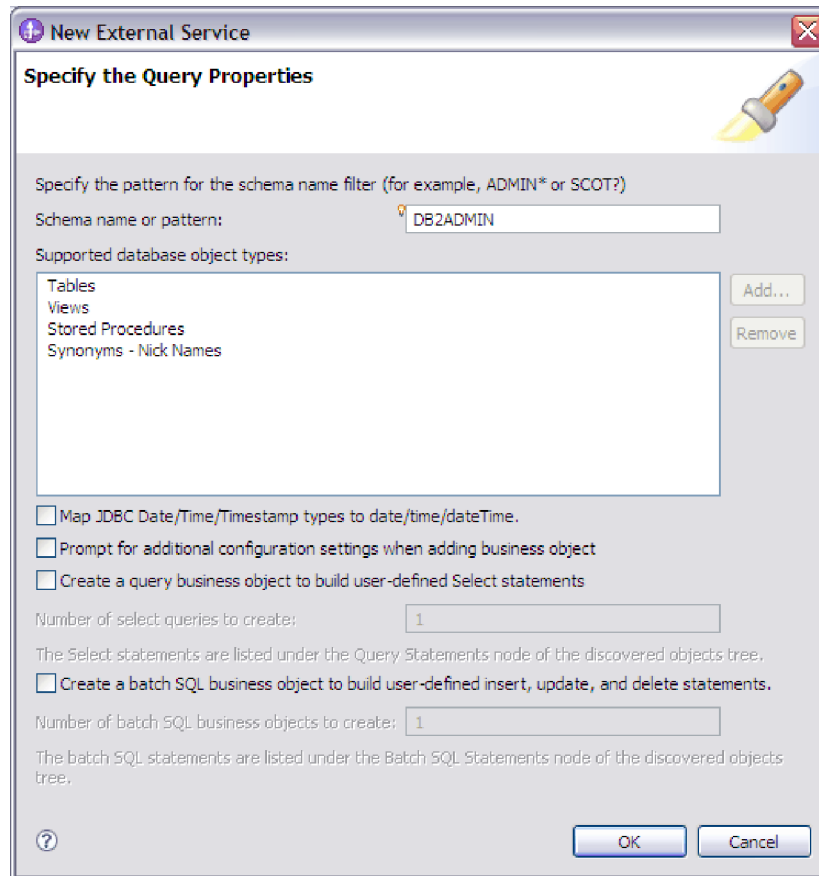
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas
- Which tables, view, synonyms or nicknames, stored procedures or stored functions you need to access
- How many query and batch SQL business objects you need to create, including parameter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

About this task

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

Procedure

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



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

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

Note: In version 6.1.x with fix pack 2, this window also allowed you to specify the number of wrapper business objects you want to create. In version 6.2.x, the wizard prompts for wrapper information at a later time.

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, stored procedures and stored functions, and synonyms or nicknames) in the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If you

need to access only specific types of database objects, omitting the ones you do not need can speed up the discovery process.

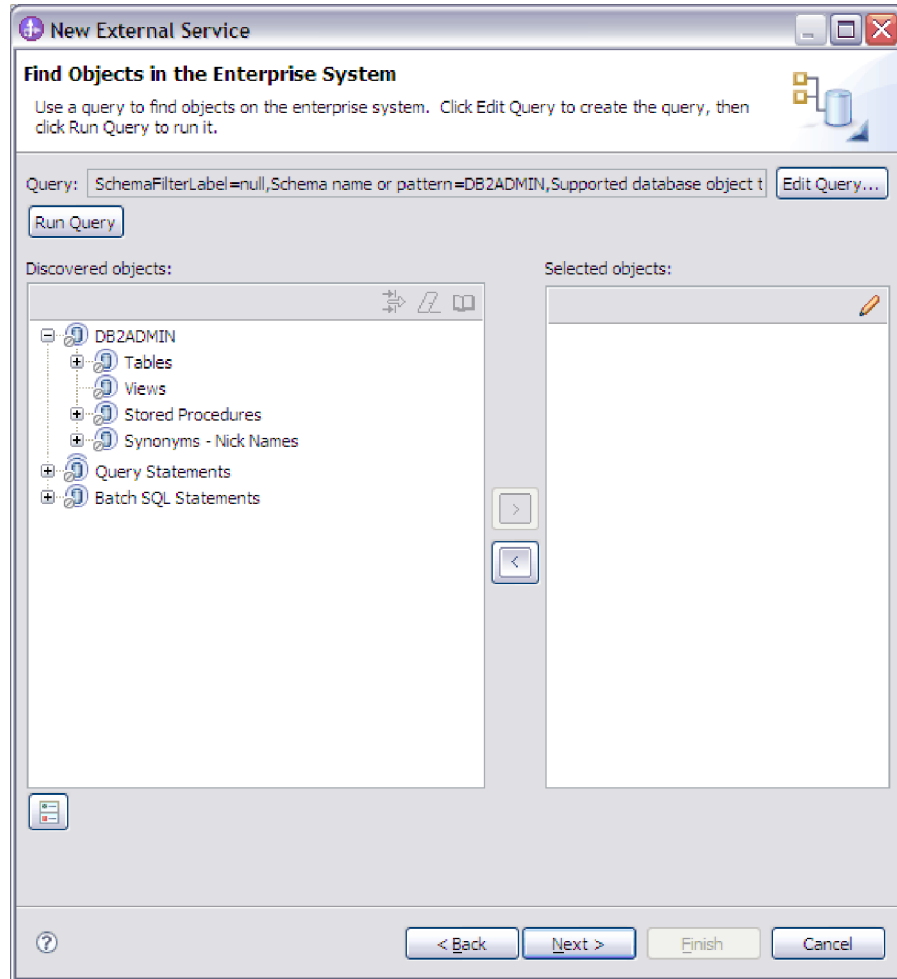
4. The table, stored procedure, and stored function objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.

Note: The default data type mapping differs based on the different JDBC driver versions, for example, when using the Oracle JDBC driver, the Date data type is mapped to dateTime data type instead of mapping it to date. In such cases, the appropriate data type must be manually selected in the Specify the Configuration Properties for 'object' window.

5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when you add a database object to the list of business objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple parent-child hierarchy of business objects. If you need a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is launched from WebSphere Integration Developer. In addition, if a foreign key reference is defined in the database, the adapter automatically discovers and displays the parent-child relationship between the tables.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor. In addition, if you have not defined a foreign key reference in the database, the adapter will not generate the parent-child relationship automatically.

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



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

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

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

Results

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

What to do next

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

Discovering database objects for inbound processing

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

Before you begin

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

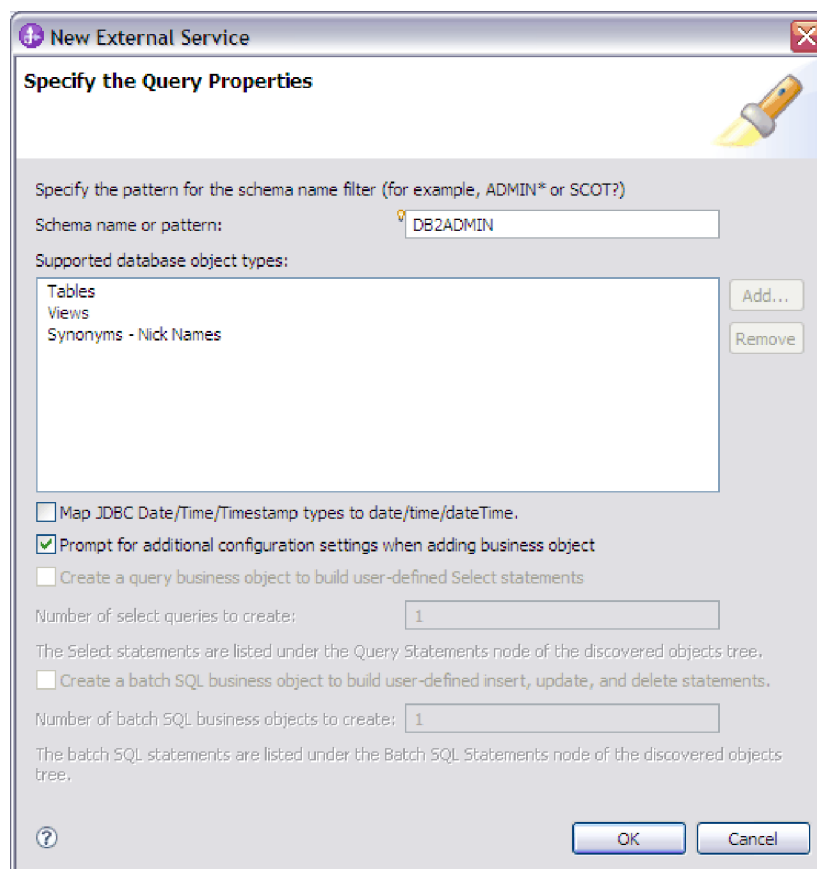
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas

About this task

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

Procedure

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



Note: The options **Create a query business object to build user-defined select statements** and **Create a batch SQL business object to build user-defined insert, update, and delete statements** are available only for outbound processing.

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

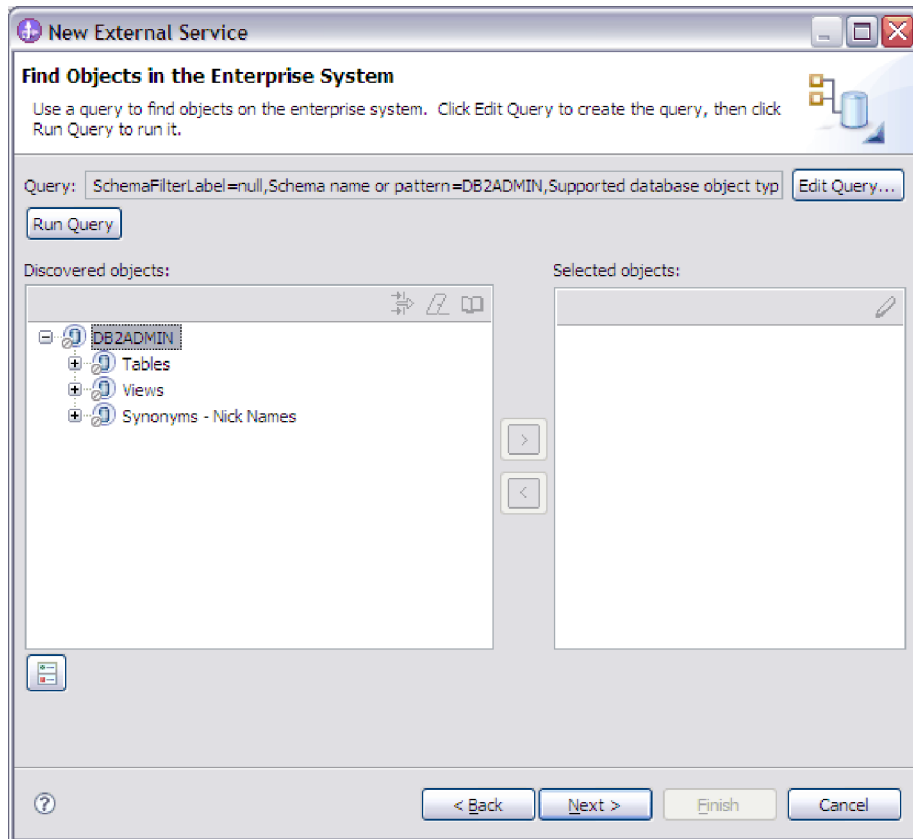
- Reduce the search time by searching a subset of database schemas
 - Omit one or more types of database objects from the search
 - Make the wizard prompt you for application-specific information that cannot be automatically determined based on information in the database
 - Map the JDBC data types Date, Time and Timestamp to date, time and dateTime
2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
 3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, and synonyms or nicknames) in the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If your database contains object types that you do not need to access, omitting them can speed up the discovery process.
 4. Table objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.

Note: The default data type mapping differs based on the different JDBC driver versions, for example, when using the Oracle JDBC driver, the Date data type is mapped to dateTime data type instead of mapping it to date. In such cases, the appropriate data type must be manually selected in the Specify the Configuration Properties for 'object' window.

5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when you add a database object to the list of business objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple parent-child hierarchy of business objects. If you need a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is launched from WebSphere Integration Developer. In addition, if a foreign key reference is defined in the database, the adapter automatically discovers and displays the parent-child relationship between the tables.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor. In addition, if you have not defined a foreign key reference in the database, the adapter will not generate the parent-child relationship automatically.

6. Click **OK** to save your changes to the query.
7. In the Find Objects in the Enterprise System window, click **Run Query** to use the query to discover database objects. The result of running a typical query is shown in the following figure.



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

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

Results

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

What to do next

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

Selecting and configuring tables, views, and synonyms or nicknames for outbound processing

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

Before you begin

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

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

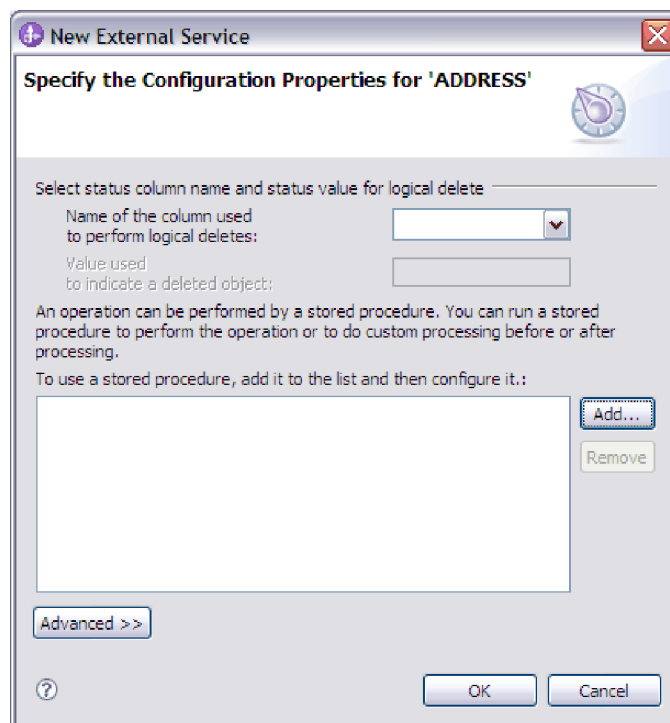
About this task

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

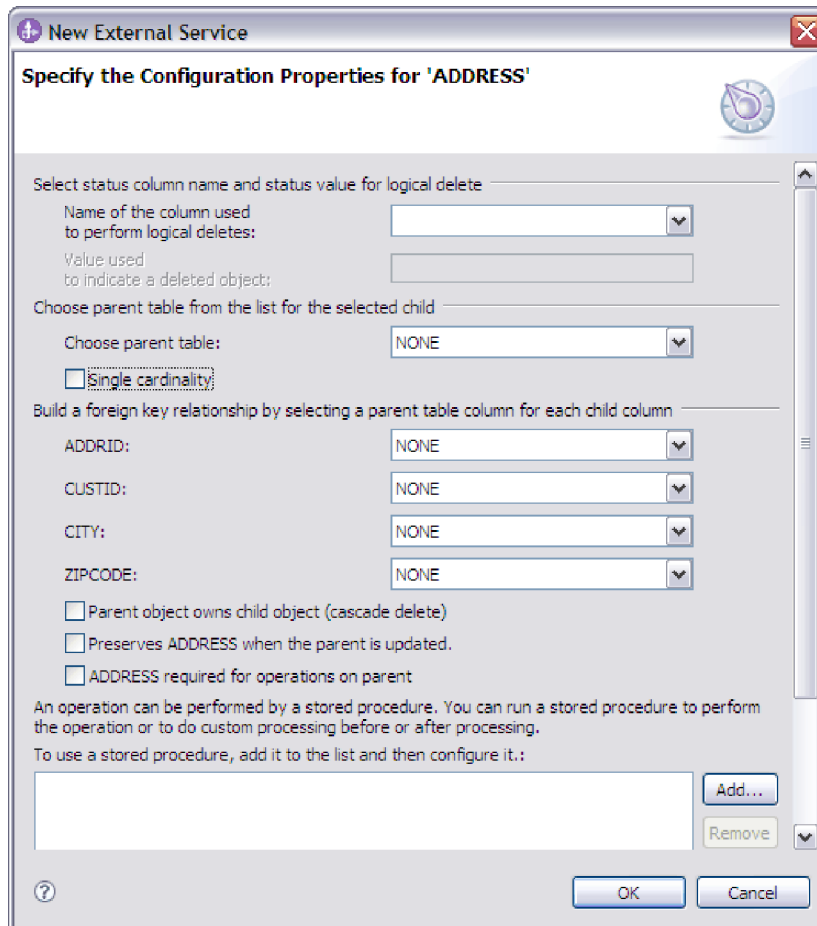
Procedure

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

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



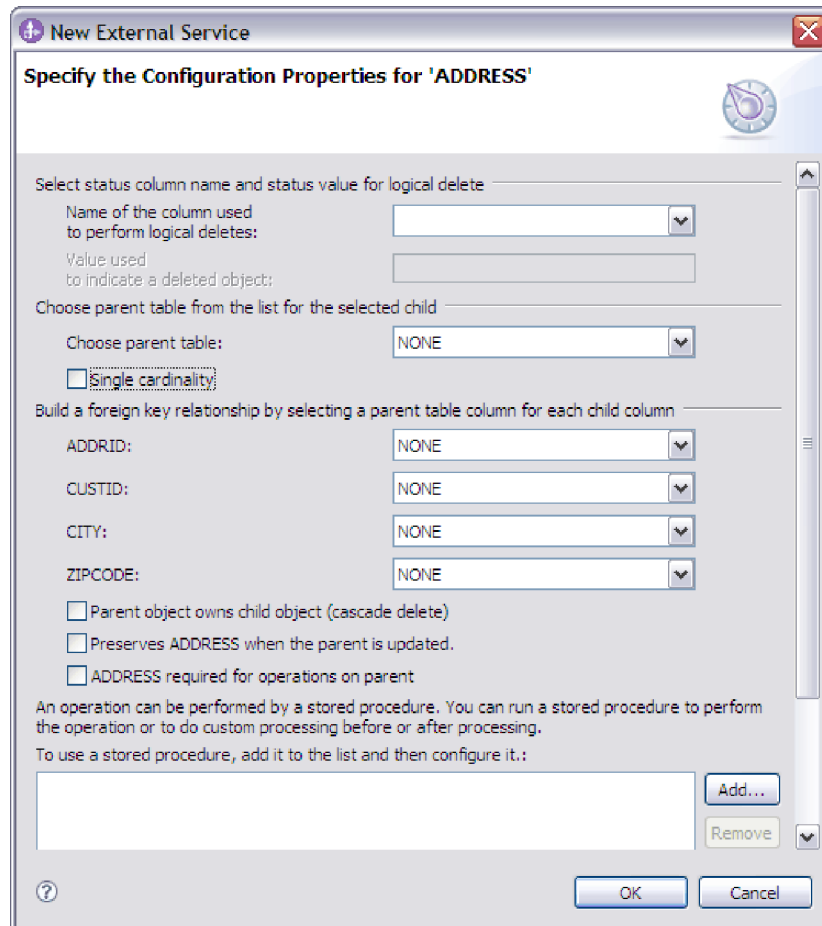
The following figure shows a typical window for subsequent tables you select. After you select and configure at least one table, the Specify the Configuration Properties for 'object' window for subsequent tables displays an area where you can optionally define a parent-child hierarchy between tables.



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

2. If the table has a column that is used to indicate logical deletes:
 - a. Select the column name in the **Name of the column used to perform logical deletes** field.
 - b. In the **Value used to indicate a deleted object** field, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. If the **Select primary key for table *table_name*** area is displayed, click **Add**, select the column to be used as the primary key for the table business object, and then click **OK**. If the table has a composite key, you can select multiple columns. The **Select primary key for table *table_name*** area is displayed only when the database table does not have a column designated as the primary key. Each table business object must have a primary key, even if the associated database table does not have a key. If the primary key is defined in the database, this section of the window is not displayed.
4. Optional: Define a parent-child relationship between business objects.
To build a parent-child hierarchy, configure the parent table first, and return to the Find Objects in the Enterprise System window to select and configure the child tables.

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



- a. In the **Choose parent table** field, select the name of the parent table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects. If you have defined a foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables after you select the parent table. If the table has a single-cardinality relationship with the parent table, the **Single cardinality** check box is automatically selected.
- b. Specify the cardinality of the relationship:
 - If the table has a single-cardinality relationship with the parent table, select the **Single cardinality** check box. In a single cardinality relationship, a parent can have only one child business object of this type. A single-cardinality relationship can be used with ownership to represent a true child or without ownership to represent lookup tables or other peer objects in a database.
 - If the table has a multiple-cardinality relationship, do not select the **Single cardinality** check box. In a multiple-cardinality relationship, a parent can have an array of child business objects of this type.
- c. Build the foreign key relationship between the parent and child by specifying for each child column whether it is a foreign key in the parent table.
 - If the child column is not a foreign key, select NONE.
 - If a child column is a foreign key, select the column in the parent table that corresponds to the child column.

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

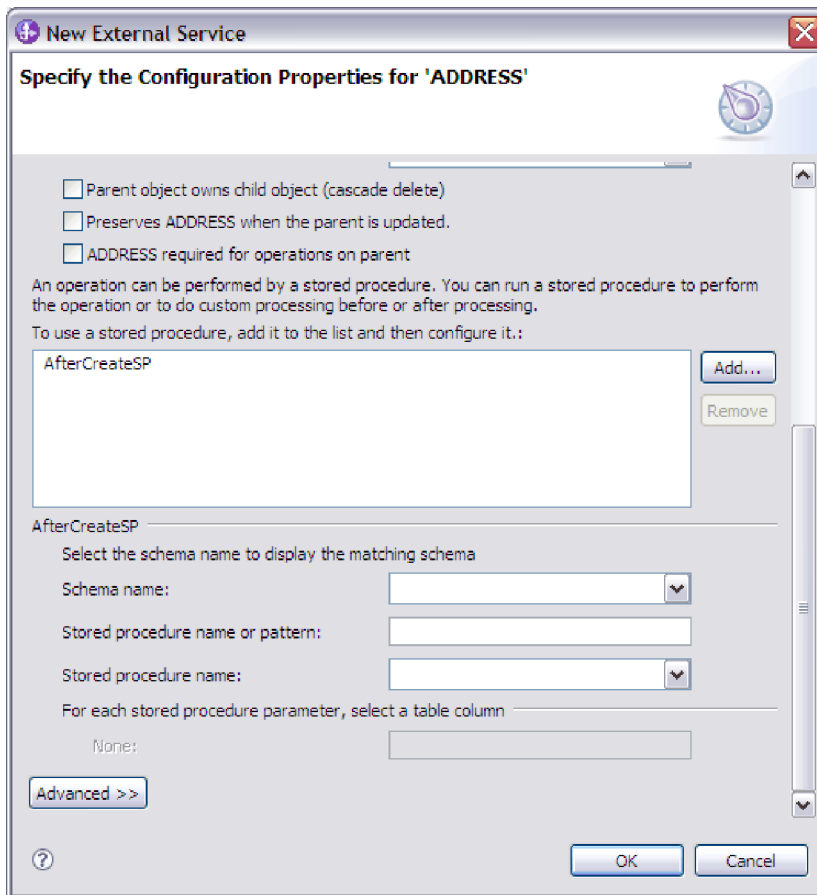
- d. If the parent object owns the child object, then the child objects in the database are deleted when the parent is deleted. To indicate that this child is owned by its parent, select the **Parent object owns child object (cascade delete)** check box. Otherwise, clear this option to prevent child objects, such as lookup tables, from being deleted when their parent is deleted.
 - e. If you do not want child objects to be deleted as part of an Update operation, select the **Preserves *child_table_name* when the parent is updated** check box.

When a parent table is updated, the adapter compares the child business objects present in the input with the child business objects returned from the database. By default, the adapter deletes any child objects returned from the database that are not present in the input business object.
 - f. By default, you can perform operations on parent business objects without specifying the child business objects. If you want to ensure that a parent business object specifies its child business objects when the parent is submitted for a change, select the ***Child_table_name* required for operations on parent** check box.
5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

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

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

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

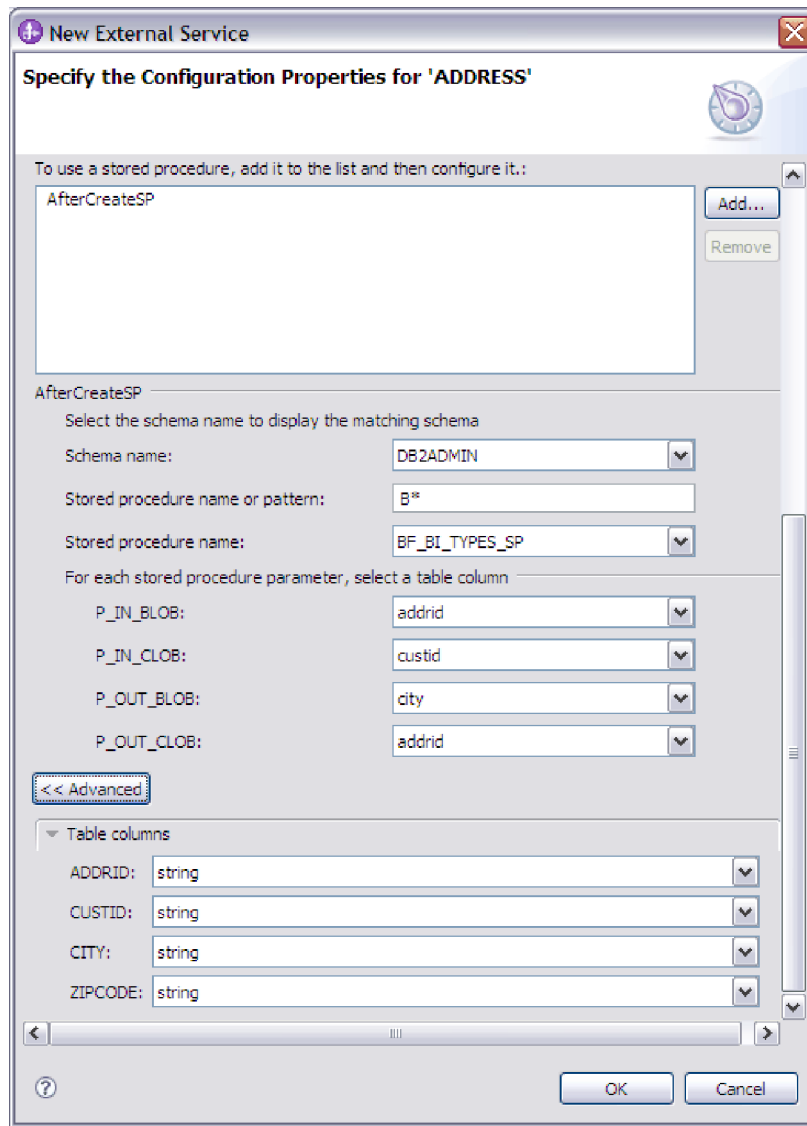


Note: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored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

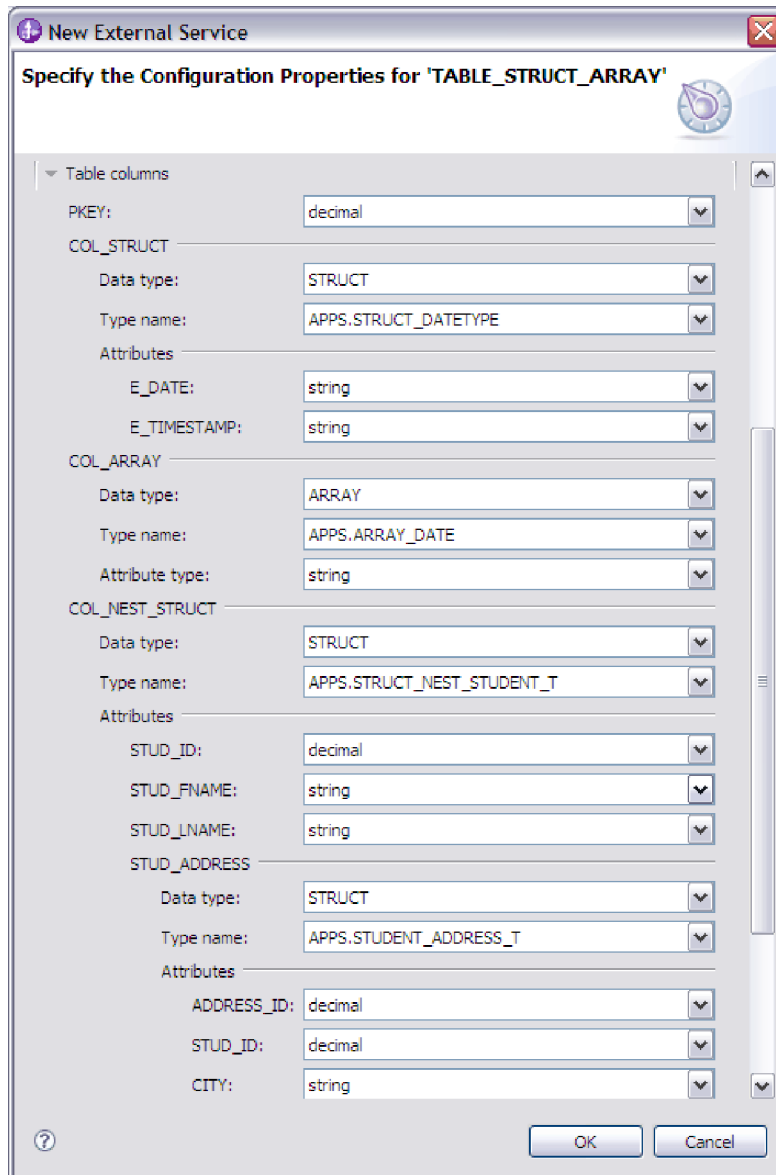
6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
 - a. In the **Schema name** field, select the name of the schema that contains the stored procedure.
 - b. Specify the name of the stored procedure or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


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

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



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



- c. Review the mapping and make changes if required.
8. When all fields in the window are completed, click **OK** to save the configuration of the business object. The table, view, synonym, and nickname business objects you defined are now listed in the Find Objects in the Enterprise System window.
9. To change the configuration of an object from the **Selected objects** list, select the object name and then click the  (Edit) icon.

What to do next

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

Selecting and configuring tables, views, and synonyms or nicknames for inbound processing

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

Before you begin

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

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

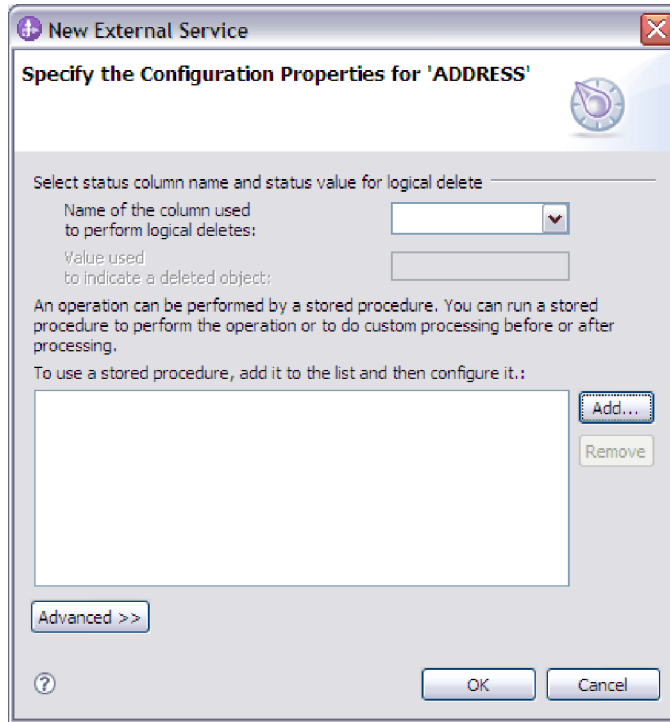
About this task

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

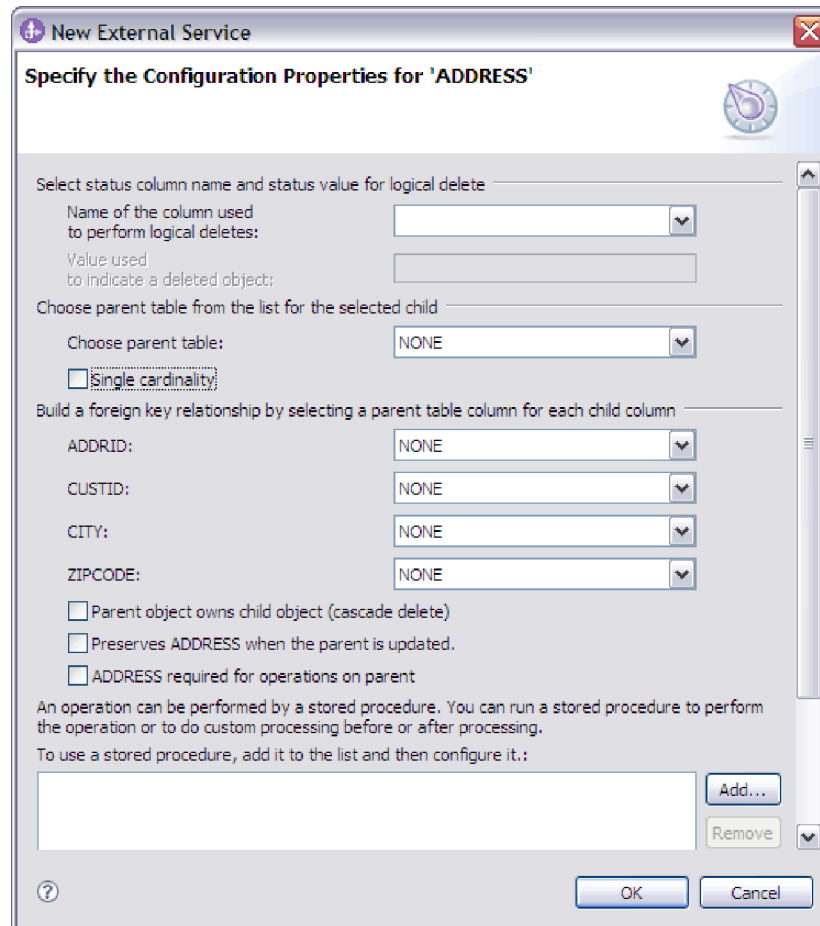
Procedure

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

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



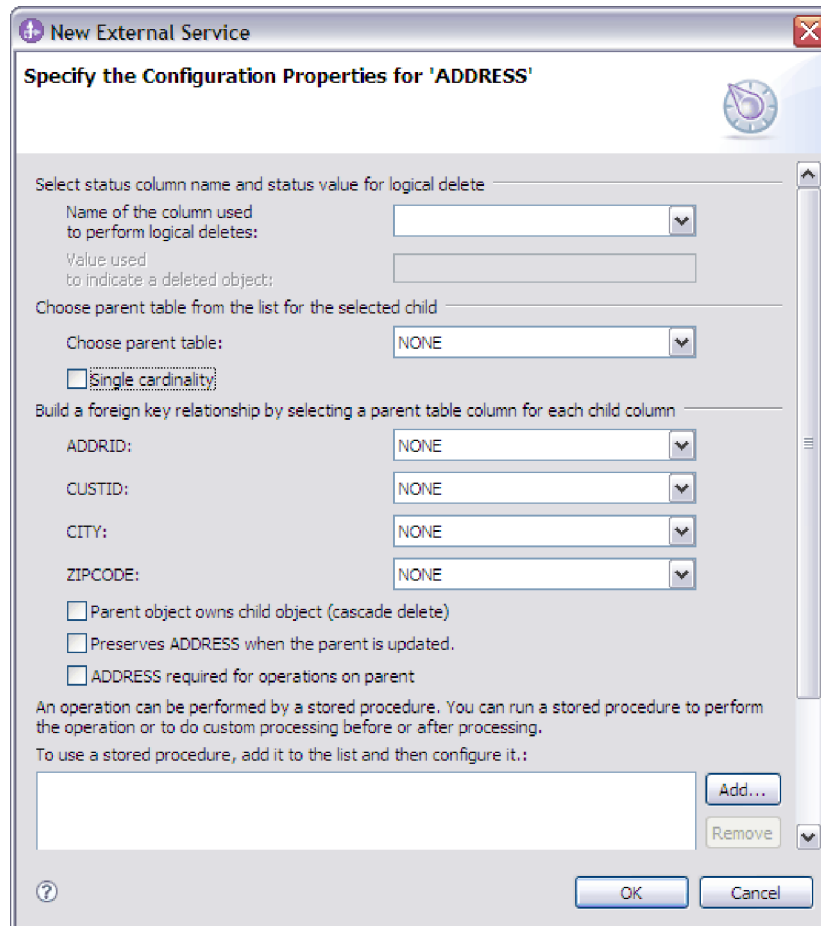
The following figure shows a typical window for subsequent tables you select. After you select and configure at least one table, the Specify the Configuration Properties for 'object' window for subsequent tables displays an area where you can optionally define a parent-child hierarchy between tables.



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

2. If the table has a column that is used to indicate logical deletes:
 - a. Select the column name in the **Name of the column used to perform logical deletes** field.
 - b. In the **Value used to indicate a deleted object** field, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. If the **Select primary key for table *table_name*** area is displayed, click **Add**, select the column to be used as the primary key for the table business object, and then click **OK**. If the table has a composite key, you can select multiple columns. The **Select primary key for table *table_name*** area is displayed only when the database table does not have a column designated as the primary key. Each table business object must have a primary key, even if the associated database table does not have a key. If the primary key is defined in the database, this section of the window is not displayed.
4. Optional: Define a parent-child relationship between business objects. To build a parent-child hierarchy, configure the parent table first, and return to the Find Objects in the Enterprise System window to select and configure the child tables.

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



- a. In the **Choose parent table** field, select the name of the parent table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects. If you have defined a foreign key reference in the database, the adapter automatically discovers and displays the parent-child relationship between the tables after you select the parent table. If the table has a single-cardinality relationship with the parent table, the **Single cardinality** check box is automatically selected.
- b. Specify the cardinality of the relationship:
 - If the table has a single-cardinality relationship with the parent table, select the **Single cardinality** check box. In a single cardinality relationship, a parent can have only one child business object of this type. A single-cardinality relationship can be used with ownership to represent a true child or without ownership to represent lookup tables or other peer objects in a database.
 - If the table has a multiple-cardinality relationship, do not select the **Single cardinality** check box. In a multiple-cardinality relationship, a parent can have an array of child business objects of this type.
- c. Build the foreign key relationship between the parent and child by specifying for each child column whether it is a foreign key in the parent table.
 - If the child column is not a foreign key, select NONE.
 - If a child column is a foreign key, select the column in the parent table that corresponds to the child column.

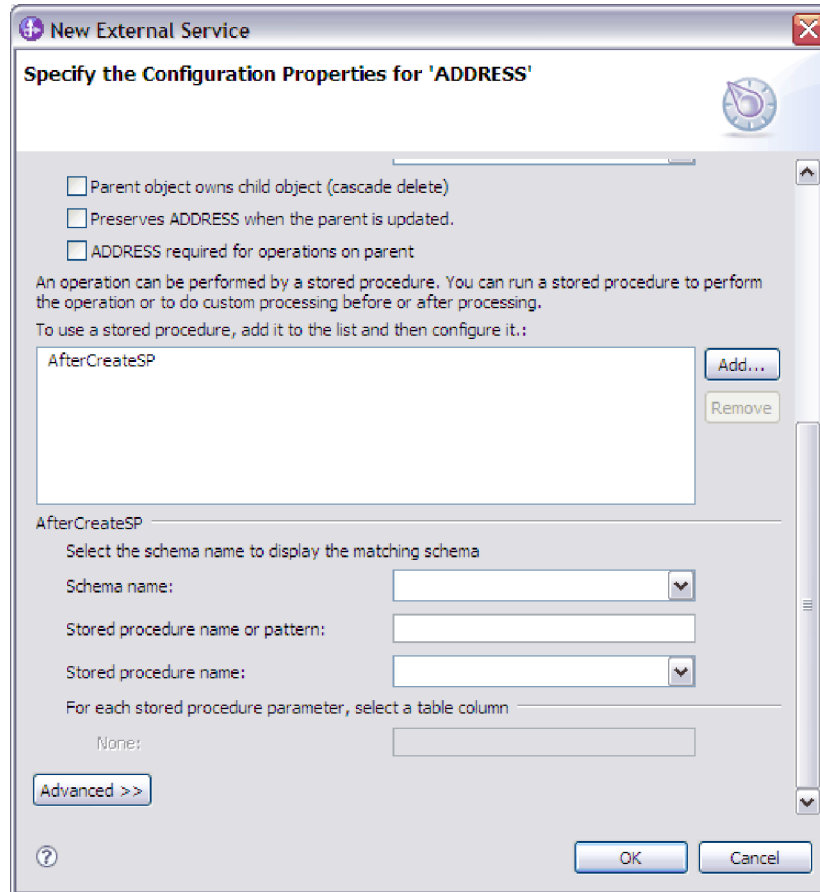
Note: The wizard can configure only a single parent table. If the child table has multiple parent tables, you must use the business object editor to configure the remaining parent tables after exiting the wizard.

- d. If the parent object owns the child object, then the child objects in the database are deleted when the parent is deleted. To indicate that this child is owned by its parent, select the **Parent object owns child object (cascade delete)** check box. Otherwise, clear this option to prevent child objects, such as lookup tables, from being deleted when their parent is deleted.
 - e. If you do not want child objects to be deleted as part of an Update operation, select the **Preserves *child_table_name* when the parent is updated** check box.
- When a parent table is updated, the adapter compares the child business objects present in the input with the child business objects returned from the database. By default, the adapter deletes any child objects returned from the database that are not present in the input business object.
- f. By default, you can perform operations on parent business objects without specifying the child business objects. If you want to ensure that a parent business object specifies its child business objects when the parent is submitted for a change, select the ***Child_table_name* required for operations on parent** check box.
5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

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

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

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

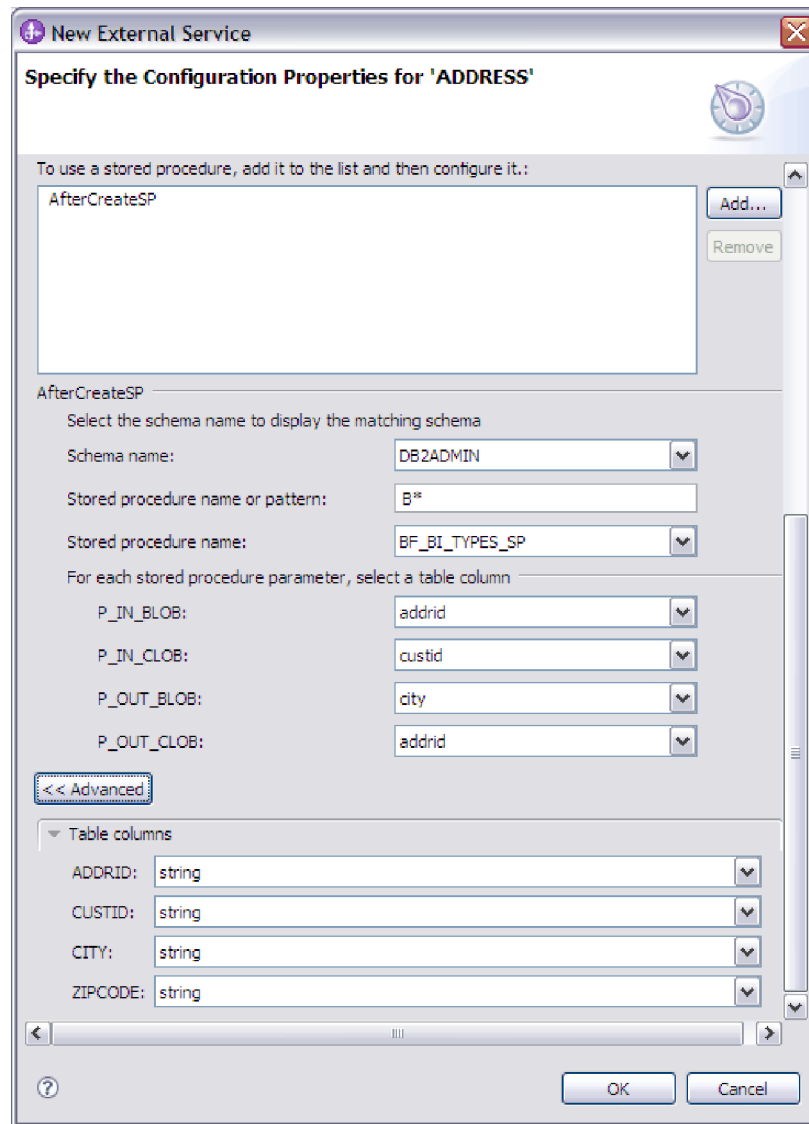


Note: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored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

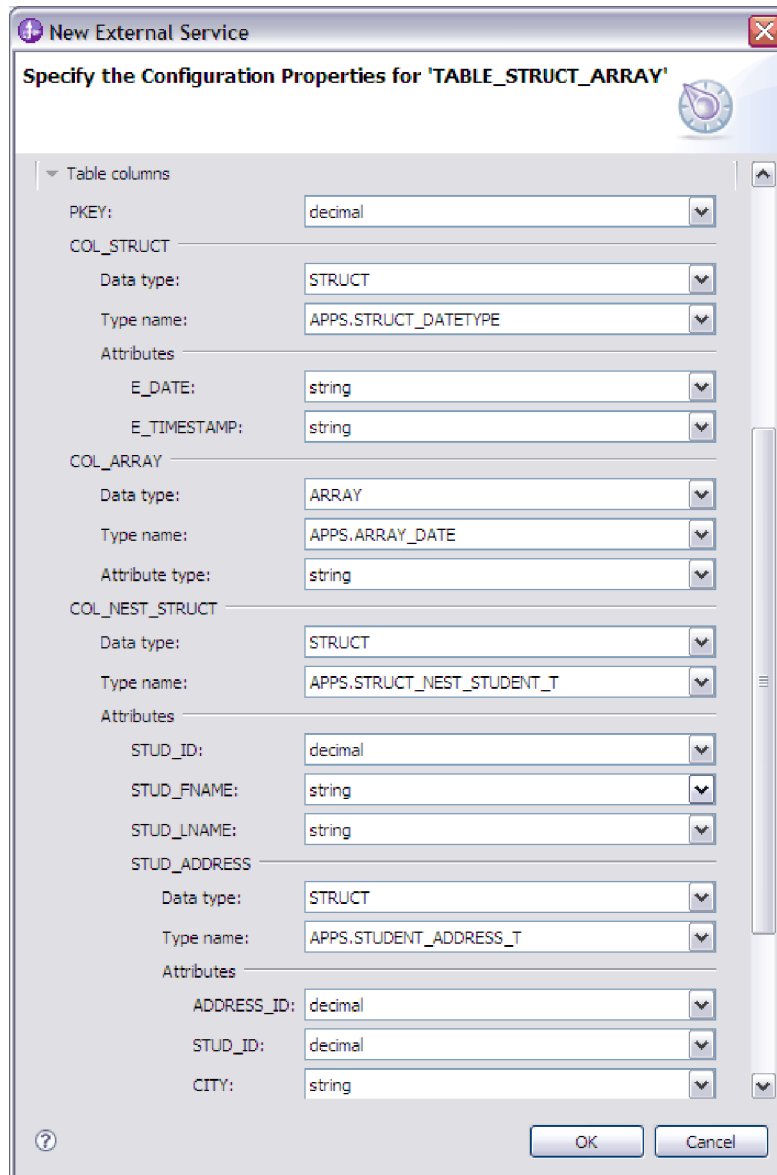
6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
 - a. In the **Schema name** field, select the name of the schema that contains the stored procedure.
 - b. Specify the name of the stored procedure or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

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

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




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



c. Review the mapping and make changes if required.

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

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

What to do next

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

global properties and configure wrapper business objects.

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](http://www.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>).



Printed in USA