Sun Adapter for SQL Server Tutorials

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–5145
07/24/2008

090714@22510

Contents

Sun Adapter for SQL Server Tutorials

The Sun Adapter for SQL Serverhandles the communication details necessary to send and receive data between these the Java CAPS environment and the SQL Server Database, and can apply business logic defined within the collaboration rules or business processes to perform data identification, manipulation, and transformation operations.

The Sun Adapter for SQL Server Tutorial provides two sample projects:

- **prjSQLServer_BPEL Sample Project**: Demonstrates how to create a Sun Business Process Manager project that uses the SQL Server Adapter. The prjSQLServer_BPEL sample project uses input files to pass data into business process. There are four business processes that demonstrate the Insert, Update, Delete, and Select operations, and one process to demonstrate a Prepared Statement. Results are written out to an output file.

  Note – You must have the eInsight.sar file installed to use the Web Services interface.

- **prjSQLServer_JCD Sample Project**: Demonstrates how to create a Java Collaboration Definition (JCD)-based project that uses the SQL Server Adapter. The prjSQLServer_JCD sample project uses input files to pass data into Collaborations. There are four Collaborations that demonstrate the Insert, Update, Delete, and Table Select operations, and one Collaboration to demonstrate a Prepared Statement. Results are written out to an output file.

It is assumed that you are already familiar with Sun Enterprise Service Bus Suite terminology and concepts.

Note – You can download the Sample Project files from `http://dscpreview.sfbay.sun.com/docs/javacaps/tutorials/index.jsp`

What You Need to Know

The following topics contain introductory and conceptual information for the Sun Adapter for SQL Tutorial.

What You Need to Do

The following topics contain the step-by-step instructions for importing and building the Sun Adapter for SQL Server sample projects.

What You Need to Do for the Java Collaboration Definition Project

The following topics contain the step-by-step instructions for importing and building the Sun Adapter for SQL Server Java Collaboration Definition Project (JCD) sample project.

What You Need to Do for the BPEL Project

The following topics contain the step-by-step instructions for importing and building the Sun Adapter for SQL Server BPEL Project sample project.

About the Sun Adapter for SQL Server Sample Projects

Topics covered in this section include:

The Sun Adapter for SQL Server Sample Project Zip file contains two sample Projects, one that is Java Collaboration Definition (JCD)-based, and one that uses the Sun Business Process Manager (BPM).

Both the `prjSQLServer_JCD` and `prjSQLServer_BPEL` sample projects demonstrate how to:

- Select employee records from a database using a prepared statement.
- Select employee records from the db_employee table.
- Insert employee records into the db_employee table.
- Update an employee record in the db_employee table.
- Delete an employee record from the db_employee table.

Sample Input Trigger Files and Output Files

In addition to the sample projects, the sql_svr.zip file also includes six sample input trigger files and ten sample output files

Sample input files

- TriggerInsert.in.~in (for JCD projects only)
- TriggerBpInsert.in.~in (for BPM projects only)
- TriggerDelete.in.~in
- TriggerUpdate.in.~in
- TriggerPsSelect.in.~in
- TriggerTableSelect.in.~in

Sample output JCD files

- JCD_Insert_output().dat
- JCD_Delete_output().dat
- JCD_Update_output().dat
- JCD_PsSelect_output().dat
- JCD_TableSelect_output().dat

Sample output BPM files

- BPEL_Insert_output().dat
- BPEL_Delete_output().dat
- BPEL_Update_output().dat
- BPEL_TableSelect_output().dat
- BPEL_PsSelect_output().dat

Sample Project Data

Data used for the sample projects are contained within a table called db_employee. The table has three columns, as follows:

TABLE 1 Sample Project Data - db_employee Table

Column Name	Data Type	Data Length
emp_no	INTEGER	10
last_name	VARCHAR	30
first_name	VARCHAR	30
rate	FLOAT	15
last_update	TIMESTAMP	19

Note – SQL Server databases do not accept table data with columns that include Boolean data types. For example, an SQL Server server expects a literal "T" or "F" for True or False Boolean values, and does not accept Java primitive Boolean True or False values.

Operations Used in the SQL Server Sample Projects

The following database operations are used in both the Java Collaboration Definition-based and Business Process Manager sample projects:

- Delete
- Insert
- Select Prepared Statement
- Select Table
- Update

About the Sun Business Process Manager Project

The prjSQLServer_BPEL sample project uses input files to pass data into business process. There are four business processes that demonstrate the Insert, Update, Delete, and Select operations, and one process to demonstrate a Prepared Statement. Results are written out to an output file.

This section contains the following topics

Associating Business Process Manager Operators

You can associate a Sun Business Process Manager Business Process Activity with the Sun Adapter for SQL Server both during the system design phase and during runtime.

To make this association:

1. Select the desired `receive` or `write` operation under the adapter in the Enterprise Explorer.
2. Drag the operation onto the eInsight Business Process canvas.

   The operation automatically changes to an Activity with an icon identifying the component that is the basis for the Activity.

The operator automatically changes to an Activity with an icon identifying the component that is the basis for the Activity.

At run time, the Sun Business Process Manager engine invokes each step in the order defined in the Business Process. Using the engine's Web Services interface, the Activity invokes the Sun Adapter for Batch.

Deploying JCD- Based Operations in the Business Process Manager

You can deploy a Java Collaboration Definition-based component as an Activity in a Sun Business Process Manager Business Process. Once you associate the desired component with an Activity, Sun Business Process Manager invokes it using a Web Services interface. Sun Enterprise Service Bus components that can interface with Sun Business Process Manager include the following:

- Java Messaging Service (JMS)
- Object Type Definitions (OTDs)
- Adapters

- Collaborations

Using the NetBeans IDE and Sun Business Process Manager, you can add an Activity to a Business Process, then associate that Activity with Sun Enterprise Service Bus component, for example, an adapter. Then, when Sun Business Process Manager runs the Business Process, it automatically invokes that component using its Web Services interface.

About the Java Collaboration Definition Sample Project

The prjSQLServer_JCD sample Project uses input files to pass data into Collaborations. There are four Collaborations that demonstrate the Insert, Update, Delete, and Table Select operations, and one Collaboration to demonstrate a Prepared Statement. Results are written out to an output file.

Assigning Operations in a Java Collaboration Definition-Based Project

Database operations are listed as methods in the JCD.

Perform the following steps to access these methods:

1. Create a Collaboration that contains an OTD using SQL Server.
2. Right-click the OTD listed in your Collaboration and then select Select Method to Call from the popup menu.
3. Browse to and select a method to call.

Importing Non-JBI Based Sample Projects

Sample projects are available for implementation and product training. You can import the sample project files from the Sun Java Caps Documentation web site at: `http://dscpreview.sfbay.sun.com/docs/javacaps/tutorials/index.jsp`.

▼ To Import a Non-JBI Based Sample Project

Before You Begin Make sure that the repository is running and that all necessary SAR files and components have been loaded. Save all unsaved work before proceeding.

1 Open your browser and locate the Sun Java CAPS Documentation web site.

2 Under the Documentation tab, click the `Tutorials` link.

3 Under `Java CAPS Tutorials and Sample Projects`, expand a tutorial list to expose the Sample Project Zip File link.

4. **Click the `Sample Project Zip File` link and extract the sample project archive file to your computer.**
   Make note of where you saved the file.

5. **Start NetBeans IDE and connect to the running repository:**

   a. **Select `Tools` → `CAPS Repository` → `Connect`.**

   b. **Supply or accept values: login name, password, etc.**

   c. **Click `Connect`.**

6. **From the NetBeans toolbar, select `Tools` → `CAPS Repository` → `Import Project`.**
   The Import Manager appears.

7. **In the Import Manager dialog box, browse to the location of the sample project archive file you extracted earlier, and select the ZIP file for the project you want to import.**

8. **In the Destination Project field, select As Top-Level.**

9. **When the sample project has successfully imported, click `Close`.**

10. **Click `Import`. The new project appears in the Projects window.**

Next Steps After you Import a project, there are several steps required to configure, deploy, and run the project.

See the following sections for directions to complete your imported project:

- "Configuring the Adapter Properties" on page 48
- "Creating the Deployment Profile" on page 52
- "Building and Deploying the Project" on page 53
- "Running the Project" on page 55

Running the SQL Script

The data used for both the Java Collaboration Definition-based and Business Process Manager sample projects are contained within a table called `db_employee`. You create this table by running the SQL statement `SQLServer_sample_script.sql`, that is included in the sample Project. Note that you must use a database tool to run the script.

The SQL statement designed for the sample projects, is as follows:

```
drop table db_employee
go
create table db_employee (
  EMP_NO int,
  LAST_NAME varchar(30),
  FIRST_NAME varchar(30),
  RATE float,
  LAST_UPDATE datetime)
go
```

The sample Projects provided with the Sun Adapter for SQL Server use input files to pass predefined data or conditions into the Collaboration or Business Process, which then transform the database contents, and deliver the ResultSet.

Creating the Java Collaboration Definition-Based Project for the Sun Adapter for SQL Server

The Java Collaboration Definition (JCD)-based project uses Business Rules defined within Java Collaborations that rely on Object Type Definitions. This section provides step-by-step instructions for manually creating the `prjSQLServer_JCD` sample Project.

The following steps are required to create the project:

Creating a New Project

First, you create a new project in the NetBeans IDE

▼ Create a Project

1 **Start the NetBeans IDE**

2 **Click the New Project icon, or select `File` → `New Project` to initiate the New Project wizard.**

3. **In Step 1 of the Wizard, select** `CAPS` → `ESB` **from the Categories column,** `CAPS Repository-Based Project` **from the Projects column, and click** `Next`**.**

4. **In Step 2 of the Wizard, specify your project name (for this project, use** `prjSQLServer_JCD` `if you are creating the Java Collaboration-based project, or prjSQLServer_BPEL if you are creating the Business Process-based project`**), and click** `Finish`**.**

Next Steps For your next step, see "Creating the OTDs" on page 13.

Creating the OTDs

The Database OTD Wizard generates Object Type Definitions (OTDs) by connecting to external data sources and creating corresponding OTDs. The sample project requires three OTDs to interact with the SQL Server Adapter.

These OTDs are:

- otdSQLServer: SQL Server Database OTD
- OTDInputDTD_DBemployees: Inbound DTD OTD
- OTDOutputDTD_DBemployees: Outbound DTD OTD

The sample project ZIP file includes DTDs used to create the inbound and outbound DTD OTDs.

▼ Create the SQL Server Database OTD

1. **Right-click your new Project in the Project window, and select** `New` → `Object Type Definition` **from the popup menu.**

   The New Object Type Definition Wizard appears.

2. **In Step 1 of the wizard, select the wizard type. In this case, select** `SQL Database` **as the wizard type. Click** `Next`**.**

3. **In Step 2 of the wizard, specify the database connection information.**

   The connection information fields are:

   - Host name
   - Port
   - Database
   - User name
   - Password

   Once you have entered the correct information in the connection fields, click `Next`.

4 **In Step 3 of the wizard, select the types of database object you want to include in your project**

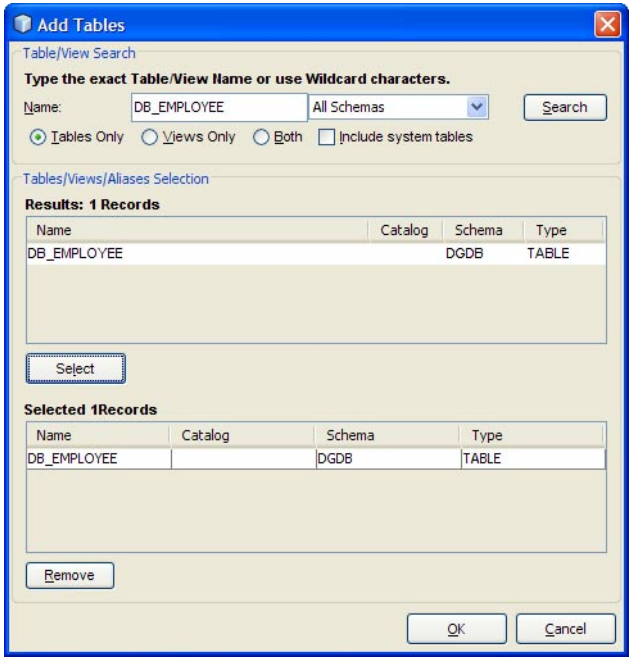
For this example, select the following:

- Tables/Views/Aliases
- Prepared Statements

Click `Next`.

5 **In Step 4 of the wizard, select the table for the project. Click `Add`. The `Add Tables` dialog box appears.**

6 **From the Add Tables dialog box, search for or enter the name of the database. For this example, use the `db_employee` table. When the database appears in the `Results` selection frame, click `Select`, and click `OK` to close the Add Table dialog box.**

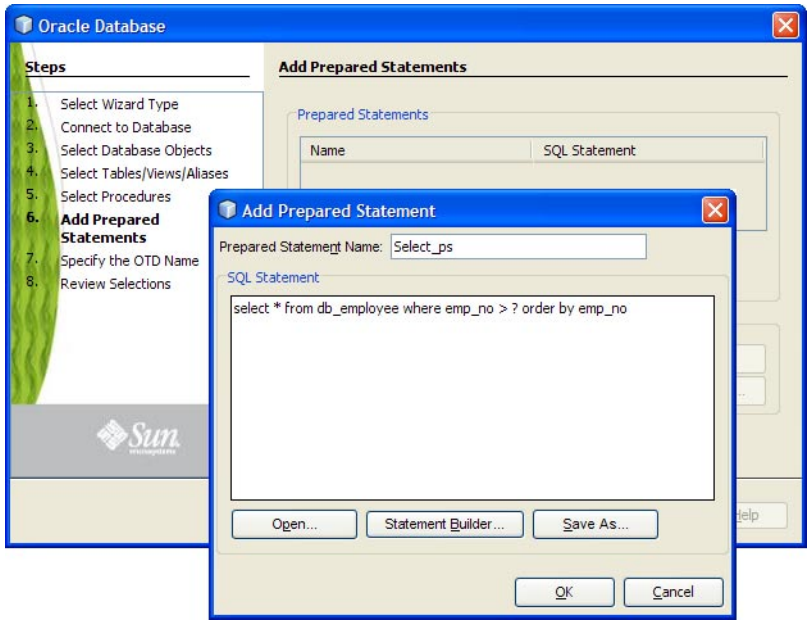


7 **Click `Next`. The wizard proceeds to Step 6, Add Prepared Statements. Click `Add`.**

The `Add Prepared Statement` dialog box appears.

8 **In the Add Prepared Statements dialog box, enter `Select_ps` as the Prepared Statement Name, and enter the following as the SQL Statement:**

```
select * from db_employee where emp_no > ? order by emp_no
```



Note – In this example, the SQL statement includes the ? placeholder for input. This placeholder represents the Where Clause.

9 **Click `OK` to close the Add Prepared Statement dialog box, and then click `Next`.**

10 **In Step 7 of the wizard, enter a name for the OTD. In this example, use otdSQLServer. Click `Next`.**

11 **In Step 8 of the wizard, review your settings, then click `Finish` to create the OTD.**

▼ Create the Inbound and Outbound DTD OTDs

1 **To create the inbound DTD OTD, right-click your project in the NetBeans Projects window, and select `New` → `Object Type Definition` from the popup menu.**

The New Object Type Definition Wizard appears.

2 **Select DTD as the wizard type, and click `Next`.**

3 **Browse to and select the `otdInputDTD.dtd` file that was extracted with the sample project. Click `Select`, and click `Next`.**

4 **In Step 3 of the wizard, select the `otdInputDTD_DBemployees` document element, and click `Next`.**

5 **In Step 4 of the wizard, accept the default option settings, and click Finish.**

The new OTD, `otdInputDTD_DBemployees`, now appears under your project's otdALL node, in the Projects window.

6 **To create the `otdOutputDTD_DBemployees` OTD, repeat steps 1–5 above, substituting otdOutputDTD.dtd for the DTD in step 3.**

When you're finished, the project's otdALL folder now contains three OTDs: `otdSQLServer`, `otdInputDTD_DBemployees`, and `otdOutputDTD_DBemployees`.

Next Steps For your next step, see:

- "Creating the Collaboration Definitions (Java)" on page 16 if you are creating the prjSQLServer_JCD Project.
- "Creating the Business Processes" on page 56 if you are creating the prjSQLServer_BPEL Project.

Creating the Collaboration Definitions (Java)

The sample project uses five Java Collaboration Definitions. Once these Collaborations are created you can write the Business Rules for each using the Collaboration Editor.

▼ Create the jcdDelete Collaboration Definition

1 **Right-click your new Project in the Project window, and select `New` → `Collaboration Definition (Java)` from the popup menu.**

The Collaboration Definition Wizard (Java) appears.

2 **Enter the name of the Collaboration (for this Collaboration enter `jcdDelete`) and click `Next`.**

3 **In Step 2 of the wizard, double-click `CAPS Components Library` → `Adapters` → `File` → `FileClient` → `receive` to add the File Adapter's receive web service operation. Click `Next`.**

4 **In Step 3 of the wizard, double-click `prjSQLServer_JCD` → `otdALL` → `otdSQLServer`. The otdSQLServer OTD is added to the Selected OTDs field.**

5 **Click the Up One Level button to return to Projects level, then select `CAPS Components Library` → `Adapters` → `File` → `FileClient` to add the File Adapter's FileClient OTD to the Collaboration.**

6 **Click Finish. The new Collaboration is added to your project's jcdALL node, in the Projects window.**

7 **Create the other four Collaborations using the Collaboration Definition Wizard, similar to the way you created the jcdDelete Collaboration, but using the following parameters:**

Collaboration Names	Web Service Operation to Implement	Selected OTDs
jcdInsert	receive (CAPS Components Library → Adapters → File → FileClient → receive)	otdSQLServer (prjSQLServer_JCD → otdALL → otdSQLServer)
		otdInputDTD_DBemployees (prjSQLServer_JCD → otdALL → otdInputDTD_DBemployees)
		FileClient (CAPS Components Library → Adapters → File → FileClient)
jcdPsSelect	receive (CAPS Components Library → Adapters → File → FileClient → receive)	otdSQLServer (prjSQLServer_JCD → otdALL → otdSQLServer)
		otdOutputDTD_DBemployees (prjSQLServer_JCD → otdALL → otdOutputDTD_DBemployees)
		FileClient (CAPS Components Library → Adapters → File → FileClient)
jcdTableSelect	receive (CAPS Components Library → Adapters → File → FileClient → receive)	otdSQLServer (prjSQLServer_JCD → otdALL → otdSQLServer)
		otdOutputDTD_DBemployees (prjSQLServer_JCD → otdALL → otdOutputDTD_DBemployees)
		FileClient (CAPS Components Library → Adapters → File → FileClient)
jcdUpdate	receive (CAPS Components Library → Adapters → File → FileClient → receive)	otdSQLServer (prjSQLServer_JCD → otdALL → otdSQLServer)
		FileClient (CAPS Components Library → Adapters → File → FileClient)

Next Steps For your next step, see "Using the Java Collaboration Editor to create Business Rules" on page 18.

Using the Java Collaboration Editor to create Business Rules

The prjSQLServer_JCD Project uses five Java Collaborations you created previously. To complete the Collaborations, use the Java Collaboration Editor's Business Rules Designer to create the business rules.

The Java Collaboration Editor also allows you to enter Java code to create business rules. The Java Source code is provided at the end of each Collaboration section, and can be copied into the Collaboration Editor's Java Source Editor to create the Collaboration.

This section contains the following topics:

Creating the Business Rules for the jcdDelete Collaboration

The jcdDelete Collaboration implements the Input Web Service Operation to read the TriggerDelete.in file and then delete the record `emp_no = 500`. The Collaboration also writes a message to JCD_Delete_output0.dat to confirm a deleted record.

Note – The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerDelete.in file is empty.

▼ Create the jcdDelete Collaboration Business Rules

1 **From the Project window, double-click the `jcdDelete` Collaboration under your project's jcdALL node.**

The Java Collaboration Editor opens to the jcdDelete Collaboration.

2 **Create the `Copy "Deleting record....."` to `FileClient_1.Text` rule.**

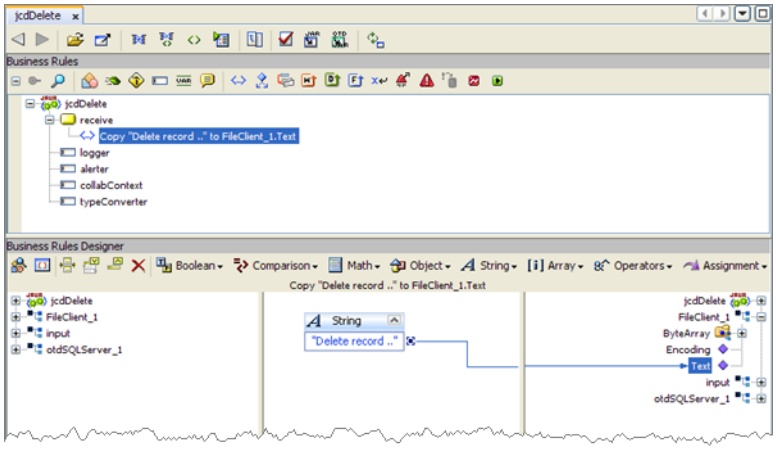
a. **From the Business Rules Designer toolbar's `String` menu, select Literal String.**

A *String* method box is added to the Business Rules Designer canvas.

b. **Double-click the value field of the *String* method box and enter `Deleting record.....` as the value.**

c. Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer. To do this, click on the output node of the *String* method box, and drag your cursor to the `Text`node, under `FileClient_1` in the right pane of the Business Rules Designer.

A visible link now connects the output node of the *String* method box and the `Text` node. The Business Rules tree now displays the new rule: `Copy "Deleting record....."
to FileClient_1.Text`.



3 Create the `FileClient_1.write` rule.

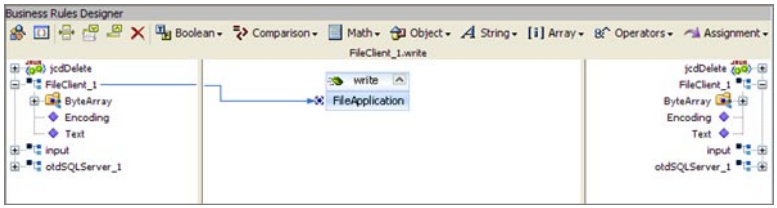
a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.

The method selection window appears.

c. Select and double-click `write()` from the method selection window.

The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.



4 Create the `otdSQLServer_1.Db_employee.delete(input.Text)` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

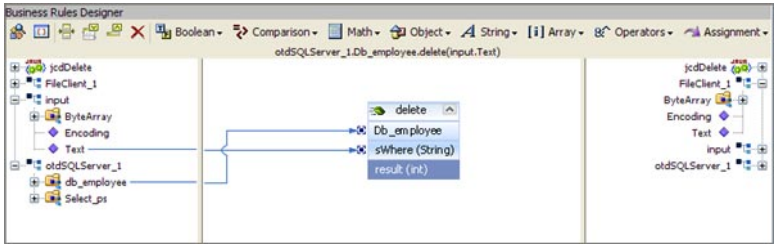
b. Right-click `Db_employee` under the otdSQLServer_1 node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `delete(StringsWhere)` from the method selection window.

The *delete* method box appears in the Business Rules Designer canvas, and a link connects the `Db_employee` node in the left pane of the Business Rules Designer to the `Db_employee` input node of the *delete* method box.

d. Map `Text` under the `input` node in the left pane of the Business Rules Designer, to the `sWhere (String)` input node of the *delete* method box.



5 Create the `Copy "Delete Done." to FileClient_1.Text` rule.

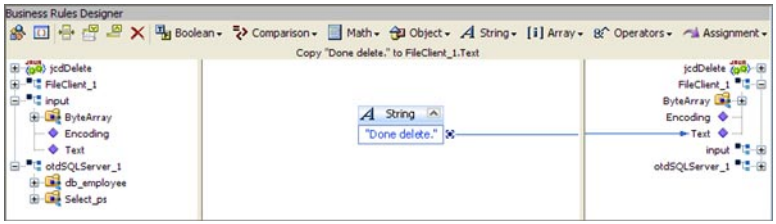
a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. From the Business Rules Designer toolbar's `String` menu, select Literal String.

A *String* method box is added to the Business Rules Designer canvas.

c. Double-click the value field of the *String* method box and enter `Delete Done` as the value.

d. Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer.



6 Create the `FileClient_1.write` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.

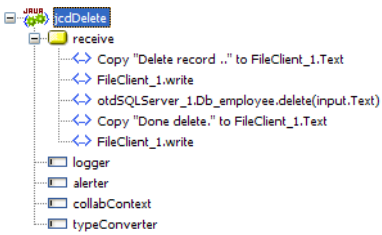
The method selection window appears.

c. Select and double-click `write()` from the method selection window.

The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

7 Click `Save All` to save your current changes.

The completed jcdDelete Collaboration definition appears as follows:



jcdDelete Collaboration Java Code

The completed Java source code for the jcdDelete Collaboration appears as follows:

```
package prjSQLServer_JCDjcdALL;
```

```
public class jcdDelete
{

   public com.stc.codegen.logger.Logger logger;

   public com.stc.codegen.alerter.Alerter alerter;

   public com.stc.codegen.util.CollaborationContext collabContext;

   public com.stc.codegen.util.TypeConverter typeConverter;

   public void receive( com.stc.connector.appconn.file.FileTextMessage input,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1, com.stc.connector.appconn.file.
FileApplication FileClient_1 )
       throws Throwable
   {
       FileClient_1.setText( "Delete record .." );
       FileClient_1.write();
       otdSQLServer_1.getDb_employee().delete( input.getText() );
       FileClient_1.setText( "Done delete." );
       FileClient_1.write();
   }

}
```

Note – The above code has been wrapped for display purposes.

Creating the Business Rules for the jcdInsert Collaboration

The jcdInsert Collaboration implements the Input Web Service Operation to read the TriggerInsert.in. file. It then unmarshals data from the input data into the otdInputDTD_DBEmployees OTD, calls the otdSQLServer OTD, and inserts records into the database via a For Loop. The Collaboration also writes a message to JCD_Insert_output0.dat to confirm an inserted record.

▼ Create the jcdInsert Collaboration Business Rules

1 **From the Project window, double-click the `jcdInsert` Collaboration under your project's jcdALL node.**

The Java Collaboration Editor opens to the jcdInsert Collaboration.

2. **Create the** `Copy "Inserting records in to db_employee table......" to FileClient_1.Text` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **From the Business Rules Designer toolbar's** `String` **menu, select** `Literal String`**.**

      A *String* method box is added to the Business Rules Designer canvas.

   c. **Double-click the value field of the** *String* **method box and enter** `Inserting records in to db_employee table......` **as the value.**

   d. **Map the output node of the** *String* **method box, to** `Text`**, under** `FileClient_1` **in the right pane of the Business Rules Designer.**

3. **Create the** `FileClient_1.write` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.**

      The method selection window appears.

   c. **Select and double-click** `write()` **from the method selection window.**

      The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

4. **Create the** `otdInputDTD_DB_Employee_1.unmarshalFromString(input.Text)` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

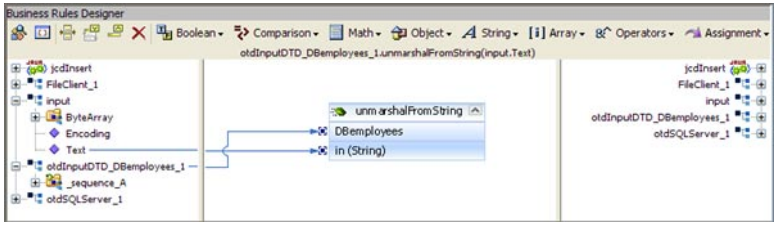
   b. **Right-click** `otdInputDTD_DB_Employee_1` **in the left pane of the Business Rules Designer, and choose** `Select method to call` **from the popup menu.**

      The method selection window appears.

   c. **Select** `unmarshalFromString(String in)` **from the method selection window.**

      The *unmarshalFromString* method box appears in the Business Rules Designer canvas, and a link connects the `otdInputDTD_DB_Employee_1` node in the left pane of the Business Rules Designer to the `Db_employee` input node of the *unmarshalFromString* method box.

d. Map `Text` under the `input` node in the left pane of the Business Rules Designer, to the `in (String)` input node of the *unmarshalFromString* method box.



5 Create the `otdSQLServerotdSQLServer_1.Db_employee.insert` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click `Db_employee` under the otdSQLServer_1 node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `insert()` from the method selection window.

The *insert* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `Db_employee` input node of the *insert* method box.

6 Create the `For Loop: i1 is less than count of otdInputDTD_DB_Employee_1.X_sequence_A` rule.

a. Click the `For Loop` icon on the Business Rules toolbar to add a For Loop to the Business Rules tree.

b. Right-click `counter initialization` node under the For Loop, and select `Local Variable` from the popup menu.

The Create Variable dialog box appears.

c. In the Create Variable dialog box, enter `i1` as the name and select `Primitive: int` as the type, and click `OK`.

The i1 variable is added to the Business Rules Designer.

d. Select `condition: ?` under the For Loop on the Business Rules tree.

e. Right-click `i1` variable node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

f. Select `Less Than` from the method selection window.

The *Less Than* method box appears in the Business Rules Designer canvas, and a link connects the `i1` variable node in the left pane of the Business Rules Designer to the `number1` input node of the *i1* method box.

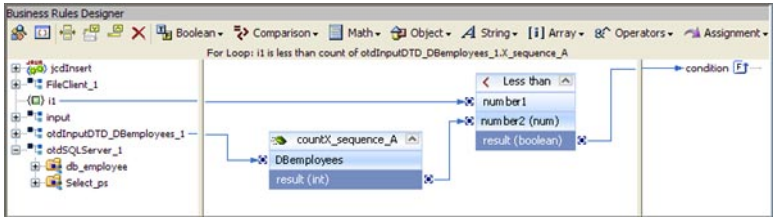
g. Right-click `otdInputDTD_DB_Employee_1` node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

h. Select `countX_sequence_A()` from the method selection window.

The *countX_sequence_A* method box appears in the Business Rules Designer canvas, and a link connects the `otdInputDTD_DB_Employee_1` node in the left pane of the Business Rules Designer to the `Db_Employee` input node of the *countX_sequence_A* method box.

i. Map the `result (int)` output node of the *countX_sequence_A* method box, to the `number2 (num)` input node of the *Less Than* method box.

j. Map the `result (boolean)` output node of the *Less Than* method box, to the For Loop `condition` node in the right pane of the Business Rules Designer.



7 Create the `Copy i1 + 1 to i1` rule beneath the For Loop → Steps node.

a. Select the `Steps` node under the For Loop and click the `rule` icon on the Business Rules toolbar to add a new rule.

b. Right-click the `i1`variable node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `Add` from the method selection window.

The *Add* method box appears in the Business Rules Designer canvas, and a link connects the `i1` node in the left pane of the Business Rules Designer to the `value1` input node of the *insert* method box.

d. Double-click the `value2` field and enter `1` as the value.

e. **Map the `result` output node of the Add method box, to the `i1` variable node in the right pane of the Business Rules Designer.**

8 **Create the `Copy Integer.parseInt(otdInputDTD_DBemployees_1.X_sequence_A[i1].EmpNo) to otdSQLServer_1.Db_employee.EMP_NO` rule beneath the For Loop → rules node.**

a. **Select the `rules` node under the For Loop in the Business Rules tree, and click the `rule` icon on the Business Rules toolbar to add a new rule.**

b. **From the Business Rules Designer toolbar, click `Class Browser`.**

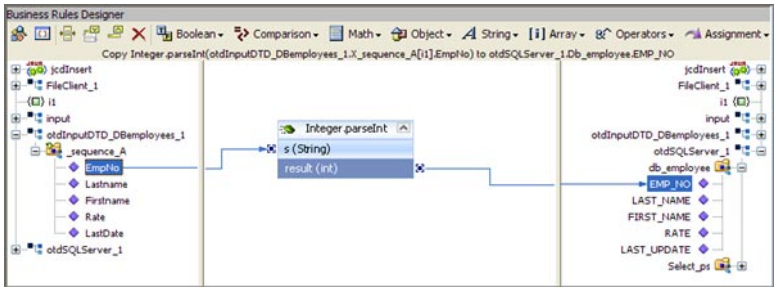
The Class Browser appears.

c. **From the Class Browser, select `Integer` as the class, and `parceInt(String s)` as the method. Click `Select`.**

The *Integer.parseInt* method box is added to the Business Rules Designer canvas.

d. **Map the `EmpNo` node, under `otdInputDTD_DBemployee_1` → `sequence_A` in the left pane of the Business Rules Designer, to the `s (String)` input node of the *Integer.parseInt* method box.**

e. **Map the result (int) output node of the *Integer.parseInt* method box, to `EMP_NO` node, under `otdSQLServer_1` `otdSQLServer_1` → `db_employee` in the right pane of the Business Rules Designer.**



9 **Create the `Copy otdInputDTD_DBemployee_1.X_sequence_A[i1].Lastname to otdSQLServer_1.db_employee.LAST_NAME` rule under the last rule.**

a. **Click the `rule` icon on the Business Rules toolbar to add a new rule.**

b. **Map the `Lastname` node, under `otdInputDTD_DBemployee_1` → `sequence_A` in the left pane of the Business Rules Designer, to the `LAST_NAME` node, under `otdSQLServer_1` `otdSQLServer_1` → `db_employee` in the right pane of the Business Rules Designer.**

10 **Create the** `Copy otdInputDTD_DBemployee_1.X_sequence_A[i1].Firstname to otdSQLServer_1.db_employee.FIRST_NAME` **rule under the last rule.**

a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

b. **Map the** `Firstname` **node, under** `otdInputDTD_DBemployee_1` → `sequence_A` **in the left pane of the Business Rules Designer, to the** `FIRST_NAME` **node, under** `otdSQLServer_1` `otdSQLServer_1` → `db_employee` **in the right pane of the Business Rules Designer.**

11 **Create the** `Copy Double.parseDouble(otdInputDTD_DBemployees_1.X_sequence_A[i1].Rate) to otdSQLServer_1.Db_employee.RATE` **rule beneath the For Loop → rules node.**

a. **Select the** `rules` **node under the For Loop in the Business Rules tree, and click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

b. **From the Business Rules Designer toolbar, click** `Class Browser`**.**

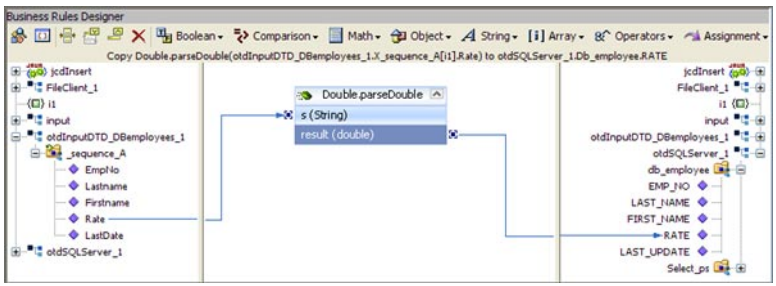
The Class Browser appears.

c. **From the Class Browser, select** `Double` **as the class, and** `parseDouble(String s)` **as the method. Click** `Select`**.**

The *Double.parseDouble* method box is added to the Business Rules Designer canvas.

d. **Map the** `Rate` **node, under** `otdInputDTD_DBemployee_1` → `sequence_A` **in the left pane of the Business Rules Designer, to the** `s (String)` **input node of the** *Double.parseDouble* **method box.**

e. **Map the** `result (double)` **output node of the** *Double.parseDouble* **method box, to the** `RATE` **node, under** `otdSQLServer_1` → `db_employee` **in the right pane of the Business Rules Designer.**



12 Create the `Copy Timestamp.valueOf(otdInputDTD_DBemployees_1.X_sequence_A[i1].LastDate) to otdSQLServer_1.Db_employee.LAST_UPDATE` rule under the last rule.

a. From the Business Rules Designer toolbar, click the `Class Browser` button.

b. From the Class Browser dialog box, select `Timestamp` as the class, and select the `valueOf(String s)` as the method. Click `Select`

A *Timestamp.valueOf* method box appears in the Business Rules Designer Canvas.

c. Map the `LastDate` node, under `otdInputDTD_DBemployee_1` → `sequence_A` in the left pane of the Business Rules Designer, to the `s (String)` input node of the *Timestamp.valueOf* method box.

d. Map the `result (Timestamp)` output node of the *Timestamp.valueOf* method box to the `LAST_UPDATE` node, under `otdSQLServer_1` `otdSQLServer_1` → `db_employee` in the right pane of the Business Rules Designer.

13 Create the `otdSQLServer_1.db_employee.insertRow` rule under the last rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click `db_employee` under the `otdSQLServer_1` node in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `insertRow()` from the method selection window.

The *insertRow* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `db_employee` input node of the *insertRow* method box.

14 Create the `Copy "Insert Done." to FileClient_1.Text` rule under the completed For LOOP.

a. From the Business Rules tree, select the `For Loop`, then click the `rule` icon on the Business Rules toolbar to add a new rule.

A new rule is added to the main trunk of the Business Rules tree.

b. From the Business Rules Designer toolbar's `String` menu, select `Literal String`.

A *String* method box is added to the Business Rules Designer canvas.

c. Double-click the value field of the *String* method box and enter `Insert Done.` as the value.

d. Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer.

15 Create the `FileClient_1.write` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.

The method selection window appears.

c. Select and double-click `write()` from the method selection window.

The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

16 Click `Save All` to save your current changes.

The completed Collaboration definition appears as follows:

jcdInsert Collaboration Java Code

The completed Java source code for the jcdInsert Collaboration appears as follows:

```
public class jcdInsert
{

   public com.stc.codegen.logger.Logger logger;

   public com.stc.codegen.alerter.Alerter alerter;
```

```
   public com.stc.codegen.util.CollaborationContext collabContext;

   public com.stc.codegen.util.TypeConverter typeConverter;

   public void receive( com.stc.connector.appconn.file.FileTextMessage
input, dtd.otdInputDTD_746620588.DBemployees otdInputDTD_DBemployees_1,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1, com.stc.connector.appconn.
file.FileApplication FileClient_1 )
       throws Throwable
   {
       FileClient_1.setText( "Inserting records into db_employee table .." );
       FileClient_1.write();
       otdInputDTD_DBemployees_1.unmarshalFromString( input.getText() );
       otdSQLServer_1.getDb_employee().insert();
       for (int i1 = 0; i1 < otdInputDTD_DBemployees_1.countX_sequence_A(); i1 += 1) {
           otdSQLServer_1.getDb_employee().setEMP_NO( Integer.parseInt
( otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getEmpNo() ) );
           otdSQLServer_1.getDb_employee().setLAST_NAME( otdInputDTD_DBemployees_1.
getX_sequence_A( i1 ).getLastname() );
           otdSQLServer_1.getDb_employee().setFIRST_NAME( otdInputDTD_DBemployees_1.
getX_sequence_A( i1 ).getFirstname() );
           otdSQLServer_1.getDb_employee().setRATE( Double.parseDouble
( otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getRate() ) );
           otdSQLServer_1.getDb_employee().setLAST_UPDATE( java.sql.Timestamp.valueOf
( otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getLastDate() ) );
           otdSQLServer_1.getDb_employee().insertRow();
       }
       FileClient_1.setText( "Done Insert." );
       FileClient_1.write();
   }

}
```

Note – The above code has been wrapped for display purposes.

Creating the Business Rules for the jcdPsSelect Collaboration

The jcdPsSelect Collaboration implements the Input Web Service Operation to read the TriggerPsSelect.in file. It then copies the database resultset (as noted in the prepared statement query) into the otdInputDTD_DBEmployee OTD and selects all available records from the database. The Collaboration also writes a message to JCD_PsSelect_output0.dat to confirm when records are selected, or when no records are available.

Using the Collaboration Editor's Java Source Editor

To create the third Java Collaboration, `jcdPsSelect`, we will use the Java Source Editor. The Java Source Editor allows you to write your business rules in the editor, or copy code into the Collaboration Editor that you have written with another tool.

jcdPsSelect Collaboration Java Code

The completed Java source code for the jcdPsSelect Collaboration appears as follows:

```
package prjSQLServer_JCDjcdALL;


public class jcdPsSelect
{

   public com.stc.codegen.logger.Logger logger;

   public com.stc.codegen.alerter.Alerter alerter;

   public com.stc.codegen.util.CollaborationContext collabContext;

   public com.stc.codegen.util.TypeConverter typeConverter;

   public void receive( com.stc.connector.appconn.file.FileTextMessage input,
dtd.otdOutputDTD_469610704.DBemployee otdOutputDTD_DBemployee_1, otdSQLServer.
OtdSQLServerOTD otdSQLServer_1, com.stc.connector.appconn.file.FileApplication
FileClient_1 )
       throws Throwable
   {
       FileClient_1.setText( "Selecting record(s) from db_employee table via
Prepared Statement select .." );
       FileClient_1.write();
       otdSQLServer_1.getSelect_ps().setEmp_no( 0 );
       otdSQLServer_1.getSelect_ps().executeQuery();
       if (otdSQLServer_1.getSelect_ps().resultsAvailable()) {
           while (otdSQLServer_1.getSelect_ps().get$Select_psResults().next()) {
               otdOutputDTD_DBemployee_1.setEmpNo( Integer.toString( otdSQLServer_1.
getSelect_ps().get$Select_psResults().getEMP_NO() ) );
               otdOutputDTD_DBemployee_1.setLastname( otdSQLServer_1.getSelect_ps().
get$Select_psResults().getLAST_NAME() );
               otdOutputDTD_DBemployee_1.setFirstname( otdSQLServer_1.getSelect_ps().
get$Select_psResults().getFIRST_NAME() );
               otdOutputDTD_DBemployee_1.setRate( Double.toString( otdSQLServer_1.
getSelect_ps().get$Select_psResults().getRATE() ) );
               otdOutputDTD_DBemployee_1.setLastDate( otdSQLServer_1.getSelect_ps().
get$Select_psResults().getLAST_UPDATE().toString() );
               FileClient_1.setText( otdOutputDTD_DBemployee_1.marshalToString() );
               FileClient_1.write();
```

```
            }
        } else {
            FileClient_1.setText( "No record found!" );
            FileClient_1.write();
        }
        FileClient_1.setText( "Done Select" );
        FileClient_1.write();
    }

}
```

Note – The above code has been wrapped for display purposes.

▼ Create the jcdPsSelect Collaboration Business Rules

1. **From the Project window, double-click the `jcdPsSelect` Collaboration under your project's jcdALL node.**

   The Java Collaboration Editor opens to the jcdPsSelect Collaboration.

2. **From the Collaboration Editor toolbar, select the Source Code Mode button.**

   The Collaboration displays the Business Rules window and the Java Source Editor window.

3. **Copy the code above and paste it into the Java Source Editor, replacing the current code. The above code was wrapped in several places for display purposes. Correct the wrapped lines of code (the wrapped lines contain no left margin) by placing your cursor at the beginning of the line and hitting backspace (or the equivalent)**

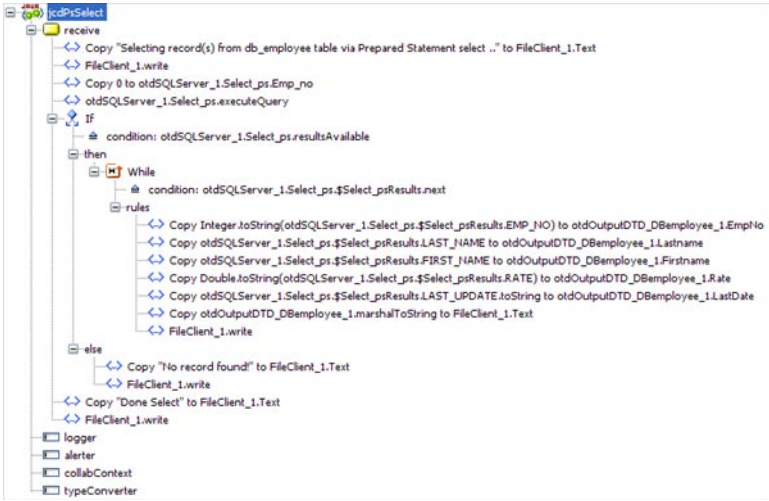
4. **Once you have corrected the code, click the `Commit changes` button on the Java Source Editor toolbar.**

   If the code contains any errors they will be listed for you in the Validation window at the bottom of the NetBeans IDE. Correct these errors if necessary, and click the `Commit changes` button again.

5. **Expand the business rules in the Business Rules window to see the completed Collaboration. Click the Advanced Mode button in the Collaboration Editor toolbar to open the Business Rules Designer window. Double click on a line of code in the Java Source Editor, and that rule is displayed graphically in the Business Rule Designer.**

6. **Click `Save All` to save your current changes.**

   The completed Collaboration definition appears as follows:



Creating the Business Rules for the jcdTableSelect Collaboration

The jcdTableSelect Collaboration implements the Input Web Service Operation to read the TriggerTableSelect.in file. It then copies the database resultset into the otdInputDTD_DBEmployee OTD and selects all available records from the database that meet the criteria `emp_no = 100`. The Collaboration also writes a message to JCD_TableSelect_output0.dat to confirm when records are selected, or when no records are available.

Note – The `where` clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.

▼ Create the jcdTableSelect Collaboration Business Rules

You can create the jcdTableSelect Java Collaboration business rules by following the steps below, or by copying the "jcdTableSelect Collaboration Java Code" on page 38 into the Collaboration Editor's Java Source Editor as described in "Using the Collaboration Editor's Java Source Editor" on page 31.

1 **From the Project window, double-click the `jcdTableSelect` Collaboration under your project's jcdALL node.**

The Java Collaboration Editor opens to the jcdTableSelect Collaboration.

2. **Create the** `Copy "Selectiong records from db_employee table via Table Select........" to FileClient_1.Text` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **From the Business Rules Designer toolbar's** `String` **menu, select** `Literal String`**.**

      A *String* method box is added to the Business Rules Designer canvas.

   c. **Double-click the value field of the *String* method box and enter** `Selectiong records from db_employee table via Table Select........` **as the value.**

   d. **Map the output node of the *String* method box, to** `Text`**, under** `FileClient_1` **in the right pane of the Business Rules Designer.**

3. **Create the** `FileClient_1.write` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.**

      The method selection window appears.

   c. **Select and double-click** `write()` **from the method selection window.**

      The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

4. **Create the** `otdInputDTD_DB_Employee_1.unmarshalFromString(input.Text)` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **Right-click** `db_employee` **under** `otdSQLServer_1` **in the left pane of the Business Rules Designer, and choose** `Select method to call` **from the popup menu.**

      The method selection window appears.

   c. **Select** `select(String where)` **from the method selection window.**

      The *select* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `Db_employee` input node of the *select* method box.

d. Map `Text` under the `input` node in the left pane of the Business Rules Designer, to the `where (String)` input node of the *select* method box.

5 Add a `While` statement and create the `condition: otdSQLServer_1.db_employee.next` rule.

a. Click the `While` icon on the Business Rules toolbar to add a While statement to the Business Rules tree.

b. Right-click `db_employee` under `otdSQLServer_1` in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `next()` from the method selection window.

The *next* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `db_employee` input node of the *next* method box.

d. Map the `result (boolean)` output node of the *next* method box, to the While `condition` in the right pane of the Business Rules Designer.

6 Create the `Copy Integer.toString(otdSQLServer_1.Db_employee.EMP_NO) to otdOutputDTD_DBemployee_1.EmpNo` rule beneath the While → rules node.

a. Select the rules node under the While statement on the Business Rules tree, and click the `rule` icon on the Business Rules toolbar to add a new rule.

b. From the Business Rules Designer toolbar, click the `Class Browser` button.

c. From the Class Browser dialog box, select `Integetr` as the class, and select the `toString(int i)` as the method. Click `Select`

A *Integer.toString* method box appears in the Business Rules Designer Canvas.

d. Map the `EMP_NO` node, under `otdSQLServer_1` → `db_employee` in the left pane of the Business Rules Designer, to the `i (int)` input node of the *Integer.toString* method box.

e. Map the `result (String)` output node of the *Integer.toString* method box to the `EmpNo` node, under `otdOutputDTD_DBemployee_1` in the right pane of the Business Rules Designer.

7 Create the `Copy otdSQLServer_1.db_employee.LAST_NAME to otdOutputDTD_DBemployee_1.Lastname` rule under the last rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule.

b. **Map the** `LAST_NAME` **node, under** `otdSQLServer_1` → `db_employee` **in the left pane of the Business Rules Designer, to the** `Lastname` **node, under** `otdOutputDTD_DBemployee_1` **in the right pane of the Business Rules Designer.**

8 **Create the** `Copy otdSQLServer_1.db_employee.FIRST_NAME to otdOutputDTD_DBemployee_1.Firstname` **rule under the last rule.**

a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

b. **Map the** `FIRST_NAME` **node, under** `otdSQLServer_1` → `db_employee` **in the left pane of the Business Rules Designer, to the** `Firstname` **node, under** `otdOutputDTD_DBemployee_1` **in the right pane of the Business Rules Designer.**

9 **Create the** `Copy Double.toString(otdSQLServer_1.db_employee.RATE) to otdOutputDTD_DBemployee_1.Rate` **rule under the last rule.**

a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

b. **Map the** `RATE` **node, under** `otdSQLServer_1` → `db_employee` **in the left pane of the Business Rules Designer, to the** `Rate` **node, under** `otdOutputDTD_DBemployee_1` **in the right pane of the Business Rules Designer.**

The Business Rules Designer adds the necessary code to create the business rule..

10 **Create the** `Copy otdSQLServer_1.db_employee.LAST_UPDATE.toString to otdOutputDTD_DBemployee_1.LastDate` **rule.**

a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

b. **Right-click** `LAST_UPDATE` **under** `otdSQLServer_1` → `db_employee` **in the left pane of the Business Rules Designer, and choose** `Select method to call` **from the popup menu.**

The method selection window appears.

c. **Select** `toString()` **from the method selection window.**

The *toString* method box appears in the Business Rules Designer canvas.

d. **Map** `LAST_UPDATE` **under** `otdSQLServer_1` → `db_employee` **in the left pane of the Business Rules Designer, to the** `Date` **input node of the** *toString* **method box.**

e. **Map the** `result (String)` **output node of the** *toString* **method box, to** `LastDate` **under** `otdOutputDTD_DBemployee_1` **in the right pane of the Business Rules Designer.**

11 Create the `Copy otdOutputDTD_DBemployee_1.marshalToString to FileClient_1.Text` rule under the last rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule.

b. Right-click `otdOutputDTD_DBemployee_1` in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.

The method selection window appears.

c. Select `marshalToString()` from the method selection window.

The *marshalToString* method box appears in the Business Rules Designer canvas, and a link connects `otdOutputDTD_DBemployee_1` in the left pane of the Business Rules Designer to the `DBemployee` input node of the *marshalToString* method box.

d. Map the `result (String)` output node of the *marshalToString* method box to the `Text` node, under `FileClient_1` in the right pane of the Business Rules Designer.

12 Create the `FileClient_1.write` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule.

b. Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.

The method selection window appears.

c. Select and double-click `write()` from the method selection window.

The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

13 Create the `Copy "Table Select Done." to FileClient_1.Text` rule.

a. From the Business Rules tree, select the `While` statement, then click the `rule` icon on the Business Rules toolbar to add a new rule.

A new rule is added to the main trunk of the Business Rules tree.

b. From the Business Rules Designer toolbar's `String` menu, select `Literal String`.

A *String* method box is added to the Business Rules Designer canvas.

c. Double-click the value field of the *String* method box and enter `Table Select Done.` as the value.

d. Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer.

14 Create the `FileClient_1.write` rule.

a. Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.

b. Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.

The method selection window appears.

c. Select and double-click `write()` from the method selection window.

The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

15 Click `Save All` to save your current changes.

The completed jcdTableSelect Collaboration definition appears as follows:

jcdTableSelect Collaboration Java Code

The completed Java source code for the jcdTableSelect Collaboration appears as follows:

```
package prjSQLServer_JCDjcdALL;


public class jcdTableSelect
{

   public com.stc.codegen.logger.Logger logger;

   public com.stc.codegen.alerter.Alerter alerter;
```

```
   public com.stc.codegen.util.CollaborationContext collabContext;

   public com.stc.codegen.util.TypeConverter typeConverter;

   public void receive( com.stc.connector.appconn.file.FileTextMessage
input, dtd.otdOutputDTD_469610704.DBemployee otdOutputDTD_DBemployee_1,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1, com.stc.connector.appconn.
file.FileApplication FileClient_1 )
       throws Throwable
   {
       FileClient_1.setText( "Selecting record(s) from db_employee table via
table select .." );
       FileClient_1.write();
       otdSQLServer_1.getDb_employee().select( input.getText() );
       while (otdSQLServer_1.getDb_employee().next()) {
           otdOutputDTD_DBemployee_1.setEmpNo( Integer.toString( otdSQLServer_1.
getDb_employee().getEMP_NO() ) );
           otdOutputDTD_DBemployee_1.setLastname( otdSQLServer_1.getDb_employee().
getLAST_NAME() );
           otdOutputDTD_DBemployee_1.setFirstname( otdSQLServer_1.getDb_employee().
getFIRST_NAME() );
           otdOutputDTD_DBemployee_1.setRate( Double.toString( otdSQLServer_1.
getDb_employee().getRATE() ) );
           otdOutputDTD_DBemployee_1.setLastDate( otdSQLServer_1.getDb_employee().
getLAST_UPDATE().toString() );
           FileClient_1.setText( otdOutputDTD_DBemployee_1.marshalToString() );
           FileClient_1.write();
       }
       FileClient_1.setText( "Done table select." );
       FileClient_1.write();
   }

}
```

Note – The above code has been wrapped for display purposes.

Creating the Business Rules for the jcdUpdate Collaboration

The jcdUpdate Collaboration implements the Input Web Service Operation to read the TriggerUpdate.in. file and then update the record `emp_no = 300`. The Collaboration also writes a message to JCD_Update_output0.dat to confirm an updated record.

Note – The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.

Create the jcdUpdate Collaboration Business Rules

You can create the jcdTableSelect Java Collaboration business rules by following the steps below, or by copying the "jcdUpdate Collaboration Java Code" on page 44 into the Collaboration Editor's Java Source Editor as described in "Using the Collaboration Editor's Java Source Editor" on page 31.

1. **From the Project window, double-click the `jcdUpdate` Collaboration under your project's jcdALL node.**

   The Java Collaboration Editor opens to the jcdUpdate Collaboration.

2. **Create the `Copy "Updating the Rate and Last_update fields .. " to FileClient_1.Text` rule.**

   a. **Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **From the Business Rules Designer toolbar's `String` menu, select `Literal String`.**

      A *String* method box is added to the Business Rules Designer canvas.

   c. **Double-click the value field of the *String* method box and enter `Updating the Rate and Last_update fields ..` as the value.**

   d. **Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer.**

3. **Create the `FileClient_1.write` rule.**

   a. **Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.**

      The method selection window appears.

   c. **Select and double-click `write()` from the method selection window.**

      The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

4 **Create the** `otdSQLServer_1.db_employee.update(input.Text)` **rule.**

   a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

   b. **Right-click** `db_employee` **under** `otdSQLServer_1` **in the left pane of the Business Rules Designer, and choose** `Select method to call` **from the popup menu.**

      The method selection window appears.

   c. **Select** `update(String sWhere)` **from the method selection window.**

      The *update* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `db_employee` input node of the *update* method box.

   d. **Map** `Text` **under the** `input` **node in the left pane of the Business Rules Designer, to the** `sWhere (String)` **input node of the** *update* **method box.**

5 **Add a** `While` **statement and create the** `condition: otdSQLServer_1.db_employee.next` **rule.**

   a. **Click the** `While` **icon on the Business Rules toolbar to add a While statement to the Business Rules tree.**

   b. **Right-click** `db_employee` **under** `otdSQLServer_1` **in the left pane of the Business Rules Designer, and choose** `Select method to call` **from the popup menu.**

      The method selection window appears.

   c. **Select** `next()` **from the method selection window.**

      The *next* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `db_employee` input node of the *next* method box.

   d. **Map the** `result (boolean)` **output node of the** *next* **method box, to the While** `condition` **in the right pane of the Business Rules Designer.**

6 **Create the Copy new Double.parseDouble("888") to otdSQLServer_1.Db_employee.RATE rule beneath the While → rules node.**

   a. **Select the** `rules` **node under the While statement in the Business Rules tree, and click the** `rule` **icon on the Business Rules toolbar to add a new rule.**

   b. **From the Business Rules Designer toolbar, click** `Class Browser`**.**

      The Class Browser appears.

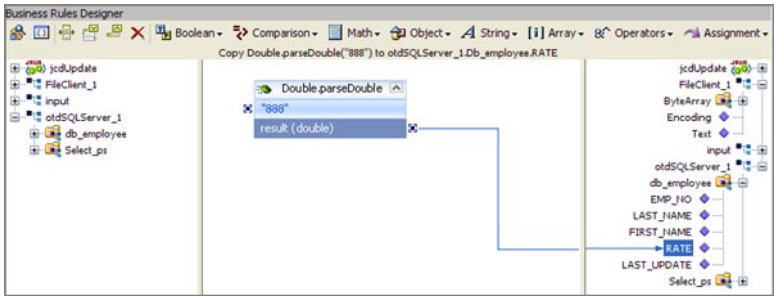
c. **From the Class Browser, select** `Double` **as the class, and** `parseDouble(String s)` **as the method. Click** `Select`**.**

The *Double.parseDouble* method box is added to the Business Rules Designer canvas.

d. **Double-click the s (String) field of the** *Double.parseDouble* **method box, and enter a value of** `888`**.**

e. **Map the** `result (Double)` **output node of the** *Double.parseDouble* **method box, to the the** `Double` **input node of the** *doubleValue* **method box.**

f. **Map the result (double) output node of the** *Double.parseDouble* **method box, to** `RATE` **node, under** `otdSQLServer_1` → `db_employee` **in the right pane of the Business Rules Designer.**



7 **Create the** `Copy Timestamp.valueOf("2008-08-08 00:00:00.0") to otdSQLServer_1.db_employee.LAST_UPDATE` **rule under the last rule.**

a. **Click the** `rule` **icon on the Business Rules toolbar to add a new rule under your last rule.**

b. **From the Business Rules Designer toolbar, click the** `Class Browser` **button.**

c. **From the ClassBrowser, select** `Timestamp` **as the class, select the**`valueOf(String s)` **method, and click** `Select`**.**

The *Timestamp.valueOf* method box appears.

d. **Double-click the** `s (String)` **field of the** *Timestamp.valueOf* **method box and enter** `2008-08-08 00:00:00.0` **as the value.**

e. **Map the** `result (Timestamp)` **output node of the** *Timestamp.valueOf* **method box to the** `LAST_UPDATE` **node, under** `otdSQLServer_1` → `db_employee` **in the right pane of the Business Rules Designer.**

8. **Create the `otdSQLServer_1.db_employee.updateRow` rule under the last rule.**

   a. **Click the `rule` icon on the Business Rules toolbar to add a new rule.**

   b. **Right-click `db_employee` in the left pane of the Business Rules Designer, and choose `Select method to call` from the popup menu.**

      The method selection window appears.

   c. **Select `updateRow()` from the method selection window.**

      The *updateRow* method box appears in the Business Rules Designer canvas, and a link connects the `db_employee` node in the left pane of the Business Rules Designer to the `db_employee` input node of the *updateRow* method box.

9. **Create the `Copy "Update Done." to FileClient_1.Text` rule.**

   a. **From the Business Rules tree, select the `While` statement, then click the `rule` icon on the Business Rules toolbar to add a new rule.**

      A new rule is added to the main trunk of the Business Rules tree.

   b. **From the Business Rules Designer toolbar's `String` menu, select `Literal String`.**

      A *String* method box is added to the Business Rules Designer canvas.

   c. **Double-click the value field of the *String* method box and enter `Update Done.` as the value.**

   d. **Map the output node of the *String* method box, to `Text`, under `FileClient_1` in the right pane of the Business Rules Designer.**

10. **Create the `FileClient_1.write` rule.**

    a. **Click the `rule` icon on the Business Rules toolbar to add a new rule in the Business Rules pane.**

    b. **Right-click the FileClient_1 node in the left pane of the Business Rules Designer, and choose Select method to call from the popup menu.**

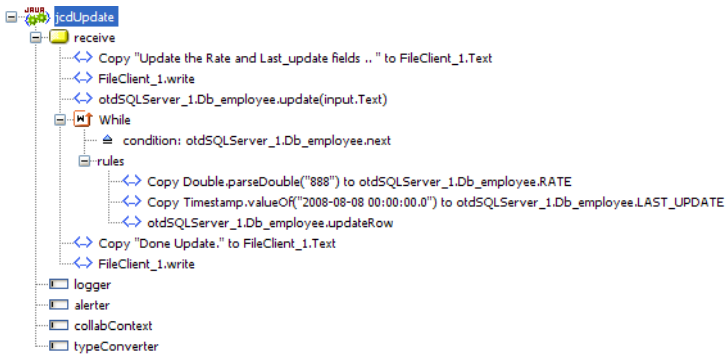
       The method selection window appears.

    c. **Select and double-click `write()` from the method selection window.**

       The *write* method box appears in the Business Rules Designer canvas. The `FileClient_1.write` rule is added to the Business Rules tree.

11. **Click `Save All` to save your current changes.**

The completed jcdUpdate Collaboration definition appears as follows:



jcdUpdate Collaboration Java Code

The completed Java source code for the jcdUpdate Collaboration appears as follows:

```
package prjSQLServer_JCDjcdALL;


public class jcdUpdate
{

   public com.stc.codegen.logger.Logger logger;

   public com.stc.codegen.alerter.Alerter alerter;

   public com.stc.codegen.util.CollaborationContext collabContext;

   public com.stc.codegen.util.TypeConverter typeConverter;

   public void receive( com.stc.connector.appconn.file.FileTextMessage input,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1, com.stc.connector.appconn.file.
FileApplication FileClient_1 )
       throws Throwable
   {
       FileClient_1.setText( "Update the Rate and Last_update fields .. " );
       FileClient_1.write();
       otdSQLServer_1.getDb_employee().update( input.getText() );
       while (otdSQLServer_1.getDb_employee().next()) {
           otdSQLServer_1.getDb_employee().setRATE( Double.parseDouble( "888" ) );
           otdSQLServer_1.getDb_employee().setLAST_UPDATE( java.sql.Timestamp.
valueOf( "2008-08-08 00:00:00.0" ) );
           otdSQLServer_1.getDb_employee().updateRow();
       }
       FileClient_1.setText( "Done Update." );
       FileClient_1.write();
```

```
    }
}
```

Note – The above code has been wrapped for display purposes.

Next Steps

For your next step, see "Creating the Connectivity Maps " on page 45.

Creating the Connectivity Maps

Connectivity Maps provide the canvas for assembling and configuring a Project's components. The Sun Adapter for SQL Server Projects use five Connectivity Maps, one for each Collaboration.

Creating a Connectivity Map involves three processes:

- Adding Connectivity Maps to a Project
- Populating the Connectivity map with the required components
- Binding the Connectivity Map Collaboration or BPEL Process to the External Applications

The Connectivity Map Designer includes a Connectivity Map Generator the can read the Collaboration Definition or Business Process. The generator then automatically populates the Connectivity map with the necessary components and creates the Bindings. We will use the Connectivity Map Generator for this sample project.

This section contains the following topics:

Adding Connectivity Maps to a Project

The SQL Server Project uses five Connectivity Maps.

▼ Add the Connectivity Maps to the Project

1 **From the NetBeans IDE Projects window, right-click your project and select** `New` → `Connectivity Map` **from the popup menu.**

The new Connectivity Map appears and adds a node on the Project tree labeled CMap1.

2 **Rename the new Connectivity Map to `cmDelete`.**

3 **Repeat this process to create the other four Connectivity Maps for your project substituting the following names:**

- cmInsert
- cmPsSelect
- cmTableSelect
- cmUpdate

Populating and Binding the Connectivity Maps using the Connectivity Map Generator

In a Connectivity Map, Adapters are associated with External Applications. For example, to establish a connection to an external SQL Server server, you must first select the SQL Server External Application to use in the Connectivity Map. Along with the External Applications, Connectivity Maps also contain Services, Queues, Topics, Web Service External Applications, and so forth. Once the Connectivity Map contains the Collaboration or Business Process and the other components, you can bind the components to create the appropriate connections.

This process can be done manually by dragging the necessary components to the Connectivity Map canvas and binding the components, or you can use the Connectivity Map Generator. The connectivity Map Generator uses the information contained in the Java Collaboration Definition or Business Process, and assembles the Connectivity Map automatically.

Note – The following steps walk you through populating the Connectivity Maps for the the prjSQLServer_JCD Project. These steps are the same for the prjSQLServer_BPEL Project. Simply substitute the appropriate Business Processes with the corresponding Java Collaborations, such as *bpDelete* for *jcdDelete*.

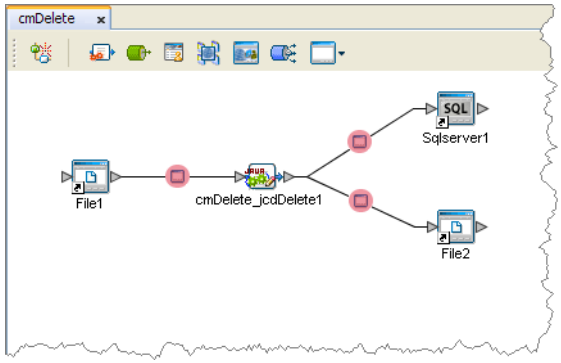
▼ Populate the Connectivity Maps using the Connectivity Map Generator.

1 **From the Project tree, double-click the cmDelete node to open the cmDelete Connectivity Map.**

2 **Drag the jcdDelete Java Collaboration onto the Connectivity Map canvas.**

3 **From the Connectivity Map toolbar, click the `Connectivity Map Generation` button.**

The Connectivity Map Generator assembles the Connectivity Map.



In the Connectivity Map, the nodes in the connections between the Collaboration and the External Applications represent the application adapters. The adapters are highlighted in red in the new Connectivity Map to indicate that the Connectivity Map properties have not been set.

4 **Save your current changes and click the** `X` **on the cmDelete tab to close the cmDelete Connectivity Map.**

5 **Repeat these steps to populate and bind the other four Connectivity Maps as follows:**

- Generate the cmInsert using the jcdInsert Collaboration
- Generate the cmPsSelect using the jcdPsSelect Collaboration
- Generate the cmTableSelect using the jcdTableSelect Collaboration
- Generate the cmUpdate using the jcdUpdate Collaboration

Next Steps For your next step, see "Creating an Environment" on page 47.

Creating an Environment

Environments include the External Systems, Logical Hosts, Application Servers, and Message Servers used by a project and contain the configuration information for these components.

▼ Create the Environment

1 **From the NetBeans IDE Services window, right-click** `Caps Environment` **and select** `New Environment` **from the popup menu.**

A new environment is added to the Services tree.

2 **Rename the new Environment to** `envSQLServerProj`**.**

3 **Right-click** `envSQLServerProj` **and select** `New` → `SQLServer External System`**. Name the SQL Server External System** `esSQLServer`**.**

4 **Right-click** `envSQLServerProj` **and select** `New` → `File External System`**. Name the File External System** `esFileClient`**.**

5 **Right-click** `envSQLServerProj` **and select** `New` → `Logical Host`**.**

LogicalHost1 is added to the Services tree.

6 **Right-click** `LogicalHost1` **and select** `New` → `Sun Java System Application Server`**.**

A new Application Server is added to the tree under LogicalHost1 .

7 **Save your changes.**

Next Steps For your next step, see "Configuring the Adapter Properties" on page 48.

Configuring the Adapter Properties

The SQL Server sample Project contains several Adapters, each represented in the Connectivity Maps as a node between an External Application and a Service. The Adapters facilitate the communication and movement of data between external applications and the Sun Enterprise Service Bus system. These Adapters must be configured for your system.

Adapter properties must be configured in both the Connectivity Maps and the Environment.

- "Configuring the Connectivity Map Properties" on page 48
- "Configuring the Environment Properties" on page 50

Configuring the Connectivity Map Properties

Connectivity Map properties are configured from each of the Connectivity Maps. These properties are specific to each of the configured Adapters, unlike the Environment properties which contain common properties for each Adapter type.

▼ Configure the Connectivity Map Properties

1 **Open a Connectivity Map, and double-click the adapter to open the Properties Editor to that adapter's property sheet. For example, open the cmDelete Connectivity Map and double-click the inbound File Adapter (the node between the File1 External Application and the cmDelete_jcdDelete1 service).**

The Properties Editor open to the selected Adapter's property sheet.

2 **From the Properties Editor, edit the values of the appropriate properties for your system.**

3 **Once you are done editing the property values for that Adapter, click** OK **to save the current properties.**

4 **Repeat this procedure to edit all of your Connectivity Map properties.**

File1 Inbound Adapter Properties

Enter the following values for the File1 Adapters.

Connectivity Map	Property Name	Required Value
cmDelete	Input file name	TriggerDelete.in
cmInsert	Input file name	TriggerBpInsert.in
cmPsSelect	Input file name	TriggerPsSelect.in
cmTableSelect	Input file name	TriggerTableSelect.in
cmUpdate	Input file name	TriggerUpdate.in

File2 Outbound Adapter Properties

Enter the following values for the outbound File2 Adapters.

Connectivity Map	Property Name	Required Value
cmDelete	Output file name	JCD_Delete_output%d.dat (for JCD Sample)
		BPEL_Delete_output%d.dat (for BPEL Sample)
cmInsert	Output file name	JCD_Insert_output%d.dat (for JCD Sample)
		BPEL_Insert_output%d.dat (for BPEL Sample)
cmPsSelect	Output file name	JCD_PsSelect_output%d.dat (JCD Sample)
		BPEL_PsSelect_output%d.dat (for BPEL Sample)
cmTableSelect	Output file name	JCD_TableSelect_output%d.dat.in (for JCD Sample)
		BPEL_TableSelect_output%d.dat.in (for BPEL Sample)
cmUpdate	Output file name	JCD_Update_output%d.dat (for JCD Sample)
		BPEL_Update_output%d.dat (for BPEL Sample)

SQL1 Outbound Adapter Properties

When you double-click the Sqlserver1 Outbound Adapter, the Adapter Connections dialog box appears. Choose an Adapter connection for your project (for this sample select `Outbound SQL Server Adapter`). When the Properties Editor for the Sqlserver1 Outbound Adapter opens, click `OK` to save the default settings.

Configuring the Environment Properties

Environment properties are configured from the Services tree of the NetBeans IDE Services window.

▼ Configure the Environment Properties

1. **From the Services tree in the NetBeans IDE Services window, expand the `CAPS Environments` → `envSQLProj` nodes**
2. **Double-click an Adapter to open the Properties Editor to the Adapter's Environment properties sheet.**
3. **Edit the property values for your system. Once you are done, click `OK` to save the current properties.**
4. **Repeat this procedure to edit all of your Environment properties.**

File Adapter Environment Properties

Enter the following values for the File Adapters.

Section	Property Name	Required Value
Configuration → Inbound File Adapter → Parameter Settings	Directory	Enter the directory that contains the input files (trigger files included in the sample Project). Trigger files: ■ TriggerDelete.in.~in ■ TriggerInsert.in.~in ■ TriggerPsSelect.in.~in ■ TriggerTableSelect.in.~in ■ TriggerUpdate.in.~in

Section	Property Name	Required Value
Configuration → Outbound File Adapter → Parameter Settings	Directory	Enter the directory where output files are written. For the JCD sample Project, the output files are: ■ JCD_Delete_output0.dat ■ JCD_Insert_output0.dat ■ JCD_PsSelect_output0.dat ■ JCD_TableSelect_output0.dat ■ JCD_Update_output0.dat For the BPEL sample Project, the output files are: ■ BPEL_Delete_output0.dat ■ BPEL_Insert_output0.dat ■ BPEL_PsSelect_output0.da ■ BPEL_TableSelect_output0.dat ■ BPEL_Update_output0.dat

SQL Server Adapter Environment Properties

Enter the following values for the outbound SQL Server Adapters.

Section	Property Name	Required Value
Configuration → Outbound SQL Server Adapter → JDBC Connector settings.	ServerName	Enter the host name of the database server being used.
	DatabaseName	Enter the name of the particular database that is being used on the server.
	User	Enter the user account name for the database.
Password	Enter the user account password for the database.	

What's the Next Step?

For your next step, see "Creating the Deployment Profile" on page 52.

Creating the Deployment Profile

A Deployment Profile is used to assign Collaborations or Business Processes and message destinations to the Sun Java System Application (GlassFish) Server and message server. Deployment Profiles are created using the Deployment Editor.

▼ Create the Deployment Profile

1 **From the NetBeans IDE Projects window, right-click the project and select** `New` → `Deployment Profile` **from the popup menu.**

2 **Enter a name for the Deployment Profile (**`dpSQLServerJCD` **for the JCD project, or** `dpSQLServerBPEL` **for the BPEL project). Click** `OK`**.**

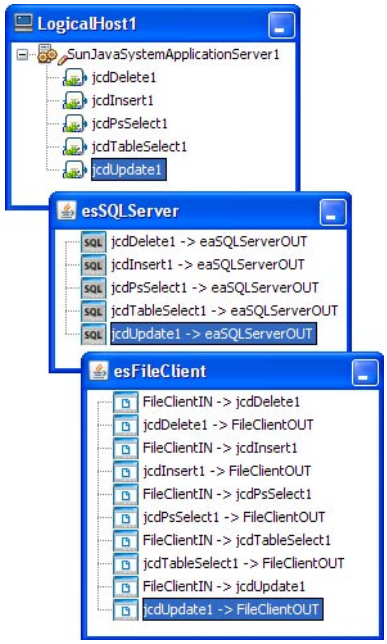
The Deployment Editor opens.

3 **From the Deployment Editor, click the** `Automap` **button.**

The Project's components are automatically mapped to their respective system windows.

Note – If any of your Project components do not successfully map to an external system, open each of your Adapter's configuration properties (Connectivity Map and Environment) and click `OK` to close and save the current configuration, then click `Automap` again.

The image below displays the Deployment Profile for the JCD sample Project.



4 **Save your project.**

Next Steps For your next step, see "Building and Deploying the Project" on page 53.

Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the project EAR file.

This section contains the following topics:

Building the Project

You can build the project from the NeBeans IDE or from the Command Line. For this example we are building the project from the Java CAPS Deployment Editor in NetBeans. For more information on building a project, see Building an Application File

▼ Build the Project

1. **From the Projects window, double-click your project's Deployment file.**
   The Deployment Editor opens to your project.
2. **From the Deployment Editor toolbar, click the `Build` button.**
3. **If there are any validation errors, they are displayed in the Validation window at the bottom of the NeBeans IDE. If there are any errors, make corrections to the project, save the project, and click the `Build` button again.**
4. **When the build succeeds, save your project.**

Deploying the Project from NetBeans

You can deploy a Java CAPS project to the Sun Java System Application Server (GlassFish) using NetBeans IDE, Admin Console, Enterprise Manager, or the Command Line. For more information on deploying projects see Deploying Java CAPS Projects.

For this example we will deploy the project from the Java CAPS Deployment Editor in NetBeans.

Note – When you deploy your project from the NetBeans IDE, you cannot specify a server instance. Therefore, if the domain has multiple server instances, the application is deployed to all of the instances.

▼ Deploy the Project

1. **Open your project in the Deployment Editor.**
2. **Ensure that the application server is running.**
3. **In the toolbar of the Deployment Profile Editor, click Deploy.**

Next Steps For your next step, see "Running the Project" on page 55.

Running the Project

When the project is running, the File Adapter polls the directory every five seconds for the input file name, as defined in the inbound File Adapter properties. The Java Collaboration or Business Process then transforms the data, and the File Adapter sends the output to an output file, as defined in the outbound File Adapter properties. See "About the Sun Adapter for SQL Server Sample Projects" on page 7 for more details on the types of output files used in this sample Project.

▼ Run the project

1 **Rename one of the trigger files included with the sample, from *filename*.in.~in to *filename*.in in the target file, to run the corresponding operation. For example, rename `TriggerDelete.in.~in` to `TriggerDelete.in` to run the *delete file* operation.**

The Where Clause defined in the business rule recognizes the trigger as a placeholder for input, allowing a set condition, such as emp_no = 100, to determine the type of output data.

You can modify the following input files to view different output.

- TriggerTableSelect.in
- TriggerDelete.in
- TriggerUpdate.in

Having no content in these files causes the operation to read all records.

2 **Verify the output data by viewing the sample output files. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.**

Creating the BPEL-Based Project for the Sun Adapter for SQL Server

The BPEL-based project uses Business Processes that rely on Business Process Execution Language (BPEL). This section provides step-by-step instructions for manually creating the `prjSQLServer_BPEL` sample Project.

The following steps are required to create the project:

- "Creating a New Project" on page 12
- "Creating the OTDs" on page 13
- "Creating the Business Processes" on page 56
- "Using Business Process Designer to Create Business Rules" on page 62
- "Creating the Connectivity Maps " on page 45

Next Steps

For your next step, see "Creating a New Project" on page 12.

Creating the Business Processes

The business process flow contains all the BPEL elements that make up a business process.

This section walks you through creating the project's five business processes:

▼ Create the bpDelete Business Process

1 **From the Projects window, right-click your new project and select New → Business Process from the shortcut menu.**

The Business Process Designer opens and BusinessProcess1 is added to the Project Explorer tree.

2 **Rename BusinessProcess1 to bpDelete.**

3 **From the Projects window, expand** `CAPS Components Library` → `Adapters` → `File` → `FileClient`**. Select and drag the** `FileClient.receive` **activity from the project tree to the BPEL Designer canvas.**

4 **As in step 3, drag the** `FileClient.write` **activity from the** `CAPS Components Library` → `Adapters` → `File` → `FileClient` **directory to the BPEL Designer canvas, and then drag another** `FileClient.write` **activity to the canvas, so that you have two FileClient.write activities in your BPEL process.**

5 **From the Projects window, expand** `prjSQLServer_BPEL` → `otdALL` → `otdSQLServer`**. Select and drag the** `otdSQLServer.Db_employeeDelete` **activity from the project tree to the BPEL Designer canvas.**

6 **Connect the activities in the bpDelete BPEL canvas by dragging your cursor from the output node of one activity to the input node of the next activity, as follows: (See image below for details)**

- Start → FileClient.receive
- FileClient.receive → FileClient.write
- FileClient.write → otdSQLServer.db_employeeDelete
- otdSQLServer.Db_employeeDelete → FileClient.write
- FileClient.write → End

7 **Click** `Save All` **to save the latest changes to your project.**

▼ Create the bpInsert Business Process

1 **From the Projects window, right-click your new project and select New → Business Process from the shortcut menu.**

2 **Rename your new business process to bpInsert.**

3 **From the Projects window, expand** `CAPS Components Library` → `Adapters` → `File` → `FileClient`**. Select and drag the** `FileClient.receive` **activity from the project tree to the BPEL Designer canvas.**

4 **As in step 3, drag the** `FileClient.write` **activity from the** `CAPS Components Library` → `Adapters` → `File` → `FileClient` **directory to the BPEL Designer canvas, and then drag another** `FileClient.write` **activity to the canvas, so that you have two FileClient.write activities in your BPEL process.**

5 **From the Projects window, expand** `prjSQLServer_BPEL` → `otdALL` → `otdInputDTD_DB_employee`**. Select and drag the** `unmarshal` **activity from the project tree to the BPEL Designer canvas.**

6 **Connect the following activities:**

- Start → FileClient.receive

- FileClient.receive → FileClient.write
- FileClient.write → otdInputDTD_DB_employee.unmarshal

7 Create the Insert Scope with a While statement.

a. From the Business Process Designer toolbar, select a `Scope` element and drag it to the canvas. Double-click the Scope element to expand the element.

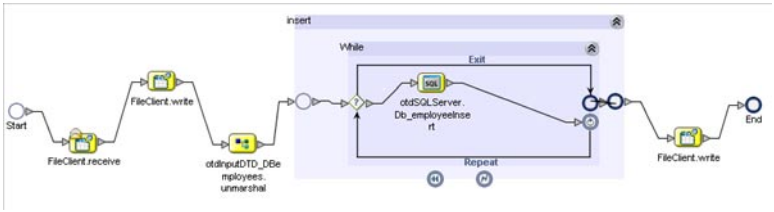
b. Rename the Scope element to `Insert`.

c. From the Business Process Designer toolbar, select a `While` element and drag it into the expanded Scope element box on the canvas. Double-click the While element to expand the element.

d. From the Projects window, expand `prjSQLServer_BPEL` → `otdALL` → `otdSQLServer`. Select and drag the `otdSQLServer.DB_EMPLOYEEInsert` activity from the project tree into the expanded While element box on the BPEL Designer canvas.

e. Connect the following elements: (See image below for details)

- otdInputDTD_DB_employee.unmarshal → Scope element input node
- Scope element input node → While element input node
- While element input node → otdSQLServer.DB_EMPLOYEEInsert
- otdSQLServer.DB_EMPLOYEEInsert → While element output node
- While element output node → Scope element output node
- Scope element output node → FileClient.write
- FileClient.write → End



8 Click `Save All` to save the latest changes to your project.

▼ Create the bpPsSelect Business Process

1 From the Projects window, right-click your new project and select New → Business Process from the shortcut menu.

2 **Rename your new business process to bpPsSelect.**

3 **Add the following activities to the bpPsSelect Business Process canvas.**

- FileClient.Receive (CAPS Components Library → Adapters → File → FileClient)
- FileClient.Write
- FileClient.Write
- Select_psPSSelectAll (prjSQLServer_BPEL → otdALL → otdSQLServer)

4 **Connect the following activities:**

- Start → FileClient.receive
- FileClient.receive → FileClient.write
- FileClient.write → otdSQLServer.Select_psPSSelectAll

5 **Create the Decision.**

a. **From the Business Process Designer toolbar, select a** `Branching Activity` → `Decision` **element and drag it to the canvas. This adds a Decision and Decision.end element to the canvas.**

b. **From the Business Process Designer toolbar, select a** `Scope` **element and drag it to the canvas. Double-click the Scope element to expand the element.**

c. **Rename the Scope element to** `Records found`**.**

d. **From the Projects window, expand** `prjSQLServer_BPEL` → `otdALL` → `otdInputDTD_DB_employee`**. Select and drag the** `marshal` **activity from the project tree into the expanded** `Records found` **element box on the BPEL Designer canvas.**

e. **From the Projects window, select and drag the** `FileClient.write` **activity from the project tree into the expanded** `Records found` **element box on the BPEL Designer canvas.**

f. **Connect the following elements: (See image below for details)**

- otdSQLServer.Select_psPSSelectAll → Decision element input node
- Decision element → Records found input node
- Records element input node → otdInputDTD_DB_employee.marshal
- otdInputDTD_DB_employee.marshal → FileClient.write
- FileClient.write → Records found element output node
- Records found element output node → Decision.end

g. **From the Business Process Designer toolbar, select another** `Scope` **element and drag it to the canvas. Double-click the Scope element to expand the element.**

h. **Rename the Scope element to** `No Record`**.**

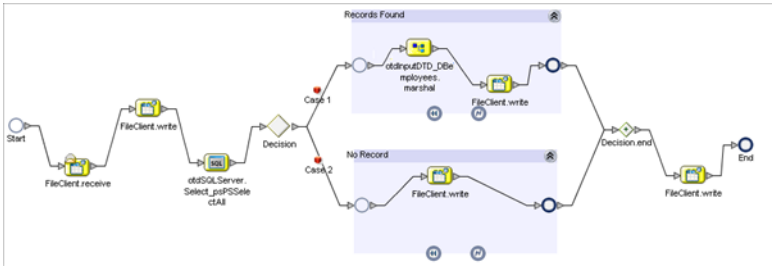
i. **From the Projects window, select and drag the `FileClient.write` activity from the project tree into the expanded `No Record` element box on the BPEL Designer canvas.**

j. **Connect the following elements: (See image below for details)**

- otdSQLServer.Select_psPSSelectAll → Decision element input node
- Decision element → No Record input node
- No Record input node → FileClient.write
- FileClient.write → No Record element output node
- No Record element output node → Decision.end

6 **Connect the following elements:**

- Decision.end → FileClient.write
- FileClient.write → End



▼ Create the bpTableSelect Business Process

1 **From the Projects window, right-click your new project and select New → Business Process from the shortcut menu.**

2 **Rename your new business process to bpTableSelect.**

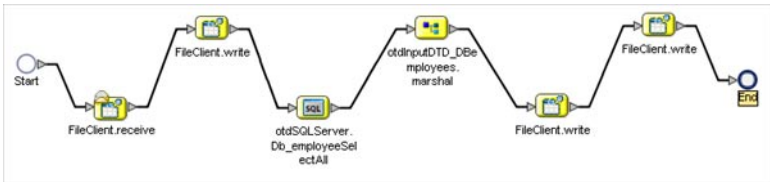
3 **Add the following activities to the bpTableSelect Business Process canvas.**

- FileClient.Receive (CAPS Components Library → Adapters → File → FileClient)
- FileClient.Write
- FileClient.Write
- FileClient.Write
- DB_EMPLOYEESelectAll (prjSQLServer_BPEL → otdALL → otdSQLServer)
- marshal (prjSQLServer_BPEL → otdALL → otdInputDTD_DB_employee)

4 **Connect the following activities: (See image below for details)**

- Start → FileClient.receive

- FileClient.receive → FileClient.write
- FileClient.write → otdSQLServer.DB_EMPLOYEESelectAll
- otdSQLServer.DB_EMPLOYEESelectAll → otdInputDTD_DB_employee.marshal
- otdInputDTD_DB_employee.marshal → FileClient.write
- FileClient.write → FileClient.write
- FileClient.write → End



Create the bpUpdate Business Process

1 From the Projects window, right-click your new project and select New → Business Process from the shortcut menu.

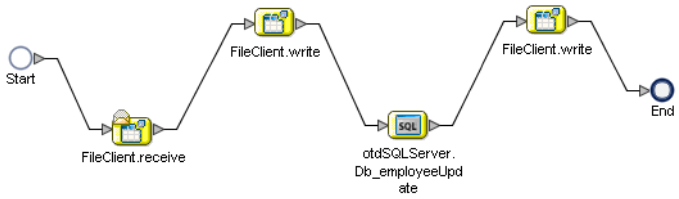
2 Rename your new business process to bpUpdate.

3 Add the following activities to the bpTableSelect Business Process canvas.

- FileClient.Receive (CAPS Components Library → Adapters → File → FileClient)
- FileClient.Write
- FileClient.Write
- DB_EMPLOYEEUpdate (prjSQLServer_BPEL → otdALL → otdSQLServer)

4 Connect the following activities: (See image below for details)

- Start → FileClient.receive
- FileClient.receive → FileClient.write
- FileClient.write → otdSQLServer.DB_EMPLOYEEUpdate
- otdSQLServer.DB_EMPLOYEEUpdate → FileClient.write
- FileClient.write → End



5 **Save your project.**

Next Steps For your next step, see "Using Business Process Designer to Create Business Rules" on page 62.

Using Business Process Designer to Create Business Rules

Business Rules, created between the Business Process Activities, allow you to configure the relationships between the input and output Attributes of the Activities using the Business Process Designer's Business Rule Designer.

This section walks you through creating the project's five business processes:

Creating the bpDelete Business Rules

The bpDelete business process describes how to delete a record in the SQL Server database using the Business Process Designer.

▼ Create the bpDelete Business Rules

1 **Double-click** `bpDelete` **in the Projects window to open the Business Process Designer to the bpDelete Business Process.**

2 **Create the** `bpDelete FileClient.receive` → `FileClient.write` **rule.**

a. **Right-click the link between** `FileClient.receive` **and** `FileClient.write` **and select** `Add Business Rule` **from the popup menu.**

A Business Rule icon is add to the link.

b. Double-click the Business Rule icon.

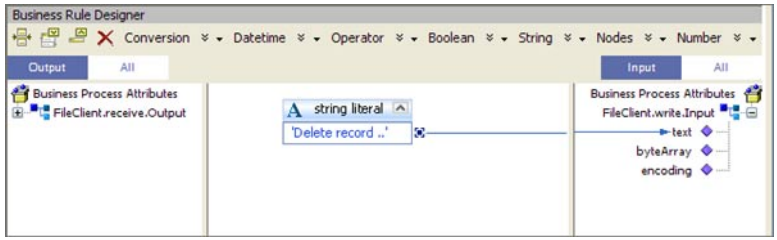
The Business Rule Designer opens to the new Business Rule.

c. From the Business Rule Designer toolbar String menu, select `string literal`.

A *string literal* method box is added to the Business Rule Designer canvas.

d. Double-click the *string literal* method box value field, and enter `Deleting record...` as the value.

e. Map the `Deleting record...` output node of the *string literal* method box, to `text` under `FileClient.write.Input` in the Input pane (right pane) of the Business Rule Designer. To do this, click on the `Deleting record...` output node of the *string literal* method box, and drag your cursor to the `text` node under `FileClient.write.Input` in the Input pane of the Business Rule Designer.



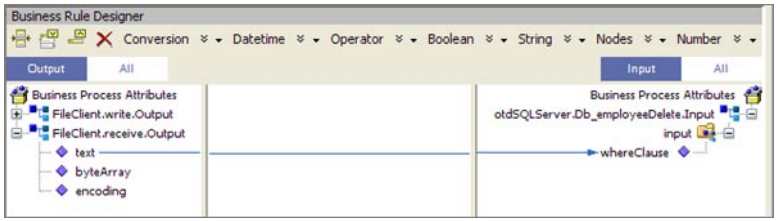
3 Create the `bpDelete` `FileClient.write` → `otdSQLServer.Db_employeeDelete` rule.

a. Add a Business Rule to the link between `FileClient.write` and `otdSQLServer.Db_employeeDelete` and double-click the new Business Rule icon.

The Business Rule Designer opens to the new Business Rule.

b. Map `text` under `FileClient.receive.Output` in the Output pane of the Business Rule Designer, to `whereClause` under `otdSQLServer.Db_employeeDelete.Input` → `input` in the Input pane of the Business Rule Designer.

A visible link now connects the two nodes in the Business Rule Designer.



4 **Create the `otdSQLServer.Db_employeeDelete` → `bpDelete` `FileClient.write` rule.**

   a. **Add a Business Rule to the link between `otdSQLServer.Db_employeeDelete` and `bpDelete` `FileClient.write` and double-click the new Business Rule icon.**

      The Business Rule Designer opens to the new Business Rule.

   b. **From the Business Rule Designer toolbar String menu, select `string literal`.**

      A *string literal* method box is added to the Business Rule Designer canvas.

   c. **Double-click the *string literal* method box value field, and enter `Delete done.` . as the value.**

   d. **Map the `Delete done.` . output node of the *string literal* method box, to `text` under `FileClient.write.Input1` in the Input pane of the Business Rule Designer.**

5 **Save your project.**

Creating the bpInsert Business Rules

The bpInsert Business Process implements the Input Operation to read the TriggerInsert.in. file. It then unmarshals data from the input data into the otdInputDTD_DBEmployees OTD, calls the otdSQLServer, and inserts records into the database, and writes a message to confirm an inserted record.

▼ Create the bpInsert Business Rules

1 **Double-click `bpInsert` in the Projects window to open the Business Process Designer to the bpInsert Business Process.**

2 **Create the `bpInsert` `FileClient.receive` → `FileClient.write` rule.**

   a. **Add a Business Rule to the link between `FileClient.receive` and `FileClient.write` and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer toolbar String menu, select `string literal`.**

c. **Enter** `Inserting records into db_employee table.` **. as the String value in the** *string literal* **method box.**

d. **Map the** `Inserting records into db_employee table.` **. output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.input` **in the Input pane (right pane) of the Business Rule Designer.**

3 **Create the** `bpInsert FileClient.write` → `otdInputDTD_DB_employee.unmarshal` **rule.**

a. **Add a Business Rule to the link between** `FileClient.write` **and** `otdInputDTD_DB_employee.unmarshal` **and double-click the new Business Rule icon.**

The Business Rule Designer opens to the new Business Rule.

b. **Map** `text` **under** `FileClient.receive.Output` **in the Output pane of the Business Rule Designer, to** `contents` **under** `otdInputDTD_DB_employee.unmarshal.Input` **in the Input pane of the Business Rule Designer.**

4 **Create the** `otdInputDTD_DB_employee.unmarshal` → `Insert (Scope) element` **rule.**

a. **Add a Business Rule to the link between** `otdInputDTD_DB_employee.unmarshal` **and the** `Insert (Scope) element` **and double-click the new Business Rule icon.**

The Business Rule Designer opens to the new Business Rule.

b. **From the Business Rule Designer toolbar Nodes menu, select** `count`**.**

A *count* method box is added to the Business Rule Designer canvas.

c. **Map the** `X_sequence_A` **under otdInputDTD_DB_employee.unmarshal.Output → DB_employee in the Output pane of the Business Rule Designer, to the** `node-set1` **input node of the** *count* **method box.**

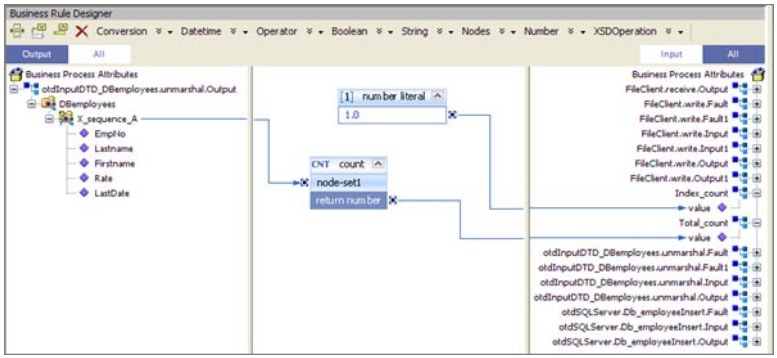
d. **Map the** `return number` **output node of the** *count* **method box, to** `value` **under** `Total_count` **in the Input pane of the Business Rule Designer.**

e. **From the Business Rule Designer toolbar Number menu, select** `number literal`**.**

A *number literal* method box is added to the Business Rule Designer canvas.

f. **Double-click the** *number literal* **method box** `value` **field, and enter** `1.0` **as the value.**

g. **Map the** `1.0` **output node of the** *number literal* **method box, to** `value` **under** `Index_count` **in the Input pane of the Business Rule Designer.**



5 Create the `X_sequence_A[number(getContainerData('Index_count'.'value'.'/value'))]` predicate for the DTD.

a. Right-click `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `x_sequence_A`, and select `New Predicate` from the popup menu.

The Predicate window appears

b. From the Predicate window's Number menu, select number.

A *number* method box is added to the Predicate window canvas.

c. Map `value` under `Index_count` in the Business Process Attributes pane of the Predicate window, to the `object1?` input node of the *number* method box.

d. Map the `Return Number` output node of the *number* method box, to `Result` in the right pane of the Predicate window.

e. Click `OK`.

The `X_sequence_A[number(getContainerData('Index_count'.'value'.'/value'))]` predicate is added to the Output pane of the Business Rule Designer under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee`.

6 Create the `While` → `otdSQLServer.DB_EMPLOYEEInsert` rule.

a. Add a Business Rule to the link between `While` input node and the `otdSQLServer.DB_EMPLOYEEInsert` and double-click the new Business Rule icon.

b. From the Business Rule Designer toolbar Number menu, select `Settings`.

The `Method Pallette` appears.

c. From the Method Palette, click the Number tab and select the `number` option. Click `Close`.

The `number` option is added to the Number menu.

d. From the Business Rule Designer toolbar Number menu, select `number`.

A *number* method box is added to the canvas.

e. Map `EmpNo` under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `X_sequence_A[number(getContainerData(´'Index_count'.'value'.'/value'))]` in the Output pane of the Business Rule Designer, to the `object1?` input node of the *number* method box.

f. Map the `return number` output node of the *number* method box, to `EMP_NO` under `otdSQLServer.DB_EMPLOYEEInsert.Input` → `input` in the Input pane of the Business Rule Designer.

g. From the Business Rule Designer toolbar Number menu, select `number` to add another *number* method box.

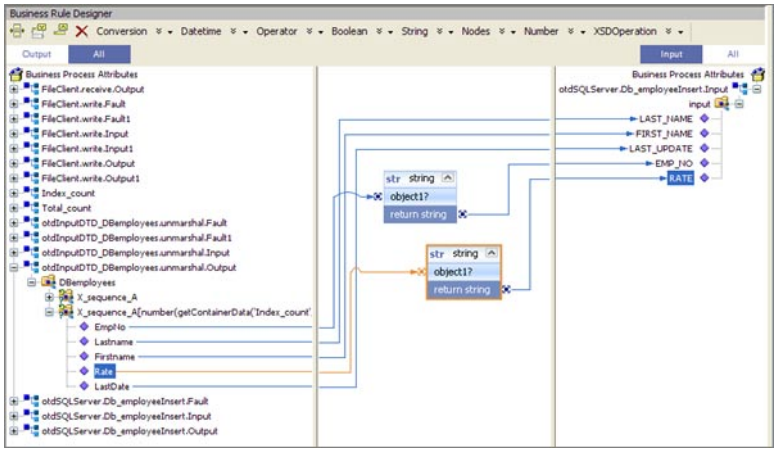
h. Map `Rate` under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `X_sequence_A[number(getContainerData(´'Index_count'.'value'.'/value'))]` in the Output pane of the Business Rule Designer, to the `object1?` input node of the *number* method box.

i. Map the `return number` output node of the *number* method box, to `RATE` under `otdSQLServer.DB_EMPLOYEEInsert.Input` → `input` in the Input pane of the Business Rule Designer.

j. Map `Lastname` under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `X_sequence_A[number(getContainerData(´'Index_count'.'value'.'/value'))]` in the Output pane of the Business Rule Designer, to `LAST_NAME` under `otdSQLServer.DB_EMPLOYEEInsert.Input` → `input` in the Input pane of the Business Rule Designer.

k. Map `Firstname` under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `X_sequence_A[number(getContainerData(´'Index_count'.'value'.'/value'))]` in the Output pane of the Business Rule Designer, to `FIRST_NAME` under `otdSQLServer.DB_EMPLOYEEInsert.Input` → `input` in the Input pane of the Business Rule Designer.

l. Map `LastDate` under `otdInputDTD_DB_employee.unmarshal.Output` → `DB_employee` → `X_sequence_A[number(getContainerData(´'Index_count'.'value'.'/value'))]` in the Output pane of the Business Rule Designer, to `LAST_UPDATE` under `otdSQLServer.DB_EMPLOYEEInsert.Input` → `input` in the Input pane of the Business Rule Designer.



7 Create the `otdSQLServer.DB_EMPLOYEEInsert` → `While` rule.

a. Add a Business Rule to the link between the `otdSQLServer.DB_EMPLOYEEInsert` activity and the `While` output node and double-click the new Business Rule icon.

b. From the Business Rule Designer toolbar Operator menu, select `addition`.

A *addition* method box is added to the canvas.

c. Map `value` under `Index_count` in the Output pane of the Business Rule Designer, to the `number1` input node of the *addition* method box.

d. Double-click the `number2` field of the *addition* method box and enter a value of `1.0`

e. Map the `return number` output node of the *addition* method box, to `value` under `Index-count` in the Input pane of the Business Rule Designer.

8 Create the `Insert` → `FileClient.write` rule.

a. Add a Business Rule to the link between the `Insert` output node and `FileClient.write` and double-click the new Business Rule icon.

b. From the Business Rule Designer toolbar String menu, select `string literal`.

c. Enter `Insert Done` as the String value.

d. Map the `Insert Done` output node of the *string literal* method box, to `text` under FileClient.write.Input1 in the Input pane of the Business Rule Designer.

Creating the bpPsSelect Business Rules

The bpPsSelect business process describes how to use a Prepared Statement query to select all records in the SQL Server database via the Business Process Designer.

▼ Create the bpPsSelect Business Rules

1 **Double-click** `bpPsSelect` **in the Projects window to open the Business Process Designer to the bpPsSelect Business Process.**

2 **Create the** `bpPsSelect FileClient.receive` → `FileClient.write` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.receive` **and** `FileClient.write` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer toolbar String menu, select** `string literal`**.**

   c. **Enter** `Selecting record(s) from db_employee table via Prepared Statement select...` **as the String value in the** *string literal* **method box.**

   d. **Map the** `Selecting record(s) from db_employee table via Prepared Statement select...` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.input` **in the Input pane (right pane) of the Business Rule Designer.**

3 **Create the** `FileClient.write` → `otdSQLServer.Select_psPSSelectAll` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.write` **and** `otdSQLServer.Select_psPSSelectAll` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer toolbar Number menu, select** `number literal`**.**

   c. **Enter** `0.0` **as the number value.**

   d. **Map the** `0.0` **output node of the** *number literal* **method box, to** `EMP_NO` **under** `otdSQLServer.Select_psPSSelectAll.Input` → `input` **in the Input pane of the Business Rule Designer.**

4 **Create the** `Records found` → `otdInputDTD_DBemployees.marshal` **rule in Case 1 of the Decision branching activity.**

   a. **Add a Business Rule to the link between the** `Records found` **input node and** `otdInputDTD_DBemployees.marshal` **and double-click the new Business Rule icon to open the Business Rule Designer.**

b. From the Business Rule Designer String menu, select `String`.

If String is not available from the String menu, click `Settings` on the String menu to open the Method Palette, and select `String` as an option from the String tab. This will add `String` to the String menu

c. Map `EMP_NO` under `otdSQLServer.Select_psPSSelectAll.Output` → `output` → `Select_psPSSelectAllPSResponseTypeList` in the Output pane of the Business Rule Designer, to the `object1?` input node of the *String* method box.

d. Map the `return string` output node of the *String* method box, to `EmpNo` under `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` in the Input pane of the Business Rule Designer

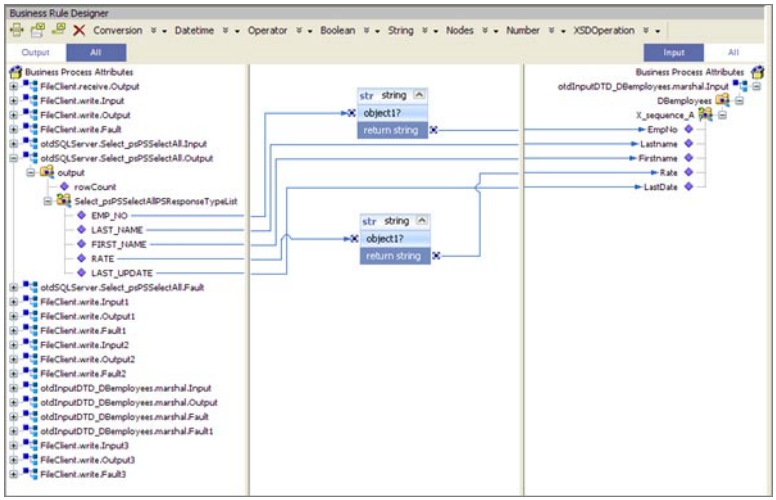
e. From the Business Rule Designer String menu, select `String` to add another *String* method box.

f. Map `RATE` under `otdSQLServer.Select_psPSSelectAll.Output` → `output` → `Select_psPSSelectAllPSResponseTypeList` in the Output pane of the Business Rule Designer, to the `object1?` input node of the *String* method box.

g. Map the `return string` output node of the *String* method box, to `Rate` under `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` in the Input pane of the Business Rule Designer

h. Map the nodes in the Output pane of the Business Rule Designer, to their corresponding nodes under `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` in the Input pane, as follows:

- LAST_NAME → Lastname
- FIRST_NAME → Firstname
- LAST_UPDATE → LastDate



5 **Create the** `otdInputDTD_DBemployees.marshal` → `FileClient.write` **rule in Case 1 of the Decision branching activity.**

   a. **Add a Business Rule to the link between** `otdInputDTD_DBemployees.marshal` **and** `FileClient.write` **within the** `Records found` **element, and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **Map** `contents` **under** `otdInputDTD_DBemployee.marshal.Output`**, in the Output pane of the Business Rule Designer, to** `text` **under** `FileClient.write.Input3` **in the Input pane of the Business Rule Designer.**

6 **Create the** `Norecord` → `FileClient.write` **rule in Case 2 of the Decision branching activity.**

   a. **Add a Business Rule to the link between** `No record` **input node and** `FileClient.write` **within the** `Records found` **element, and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer toolbar String menu, select** `string literal`**.**

   c. **Enter** `Records Not Found` **as the String value in the** *string literal* **method box.**

   d. **Map the** `Records Not Found` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.Input2` **in the Input pane (right pane) of the Business Rule Designer.**

7 Create the Decision Gate Properties for the Decision Case 1 and Case 2.

a. Double-click the red `Case 1` icon between `Decision` input node and `Records found` input node.

The Decision Gate Properties Editor appears.

b. From the Decision Gate Properties Editor toolbar Operator menu, select `Settings`.

The `Method Pallette` appears.

c. From the Method Palette, click the Operator tab and select the `greater than` option (for Case 1) and select the `lesser or equal` option (for case 2). Click `Close`.

The new options are added to the Operator menu.

d. From the Decision Gate Properties Editor toolbar Operator menu, select `greater than`.

The *greater than* method box is added to the editor's canvas.

e. Map `rowCount` under `otdSQLServer.Select_psPSSelectAll.Output` → `output` in the left pane of the Decision Gate Properties Editor, to the `number1` input node of the *greater than* method box.

f. Double-click the `number2` field of the *greater than* method box, and change the value to `0.0`.

g. Map `00` output node of the *greater than* method box, to the Result node in the right pane of the Decision Gate Properties Editor.

h. From the Order of Execution field, select `Case 2`.

i. From the Decision Gate Properties Editor toolbar Operator menu, select `lessor or equal`.

The *lessor or equal* method box is added to the editor's canvas.

j. Map `rowCount` under `otdSQLServer.Select_psPSSelectAll.Output` → `output` in the left pane of the Decision Gate Properties Editor, to the `number1` input node of the *lessor or equal* method box.

k. Double-click the `number2` field of the *lessor or equal* method box, and change the value to `0.0`.

l. Map `00` output node of the *lessor or equal* method box, to the Result node in the right pane of the Decision Gate Properties Editor.

m. Click OK to close the Decision Gate Properties Editor.

The Case 1 and 2 icons change from red to green to indicate that the properties have been set.

8. **Create the** `Decision.end` → `FileClient.write` **rule.**

   a. **Add a Business Rule to the link between the** `Decision.end` **element and** `FileClient.write`**, and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer toolbar String menu, select** `string literal`**.**

   c. **Enter** `Select Done` **as the String value in the** *string literal* **method box.**

   d. **Map the** `Select Done` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.Input3` **in the Input pane of the Business Rule Designer.**

9. **Save your project.**

Creating the bpTableSelect Business Rules

The bpTableSelect business process describes how to select all records the SQL database using the Business Process Designer.

Note – The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.

▼ Create the bpTableSelect Business Rules

1. **Double-click** `bpTableSelect` **in the Projects window to open the Business Process Designer to the bpTableSelect Business Process.**

2. **Create the** `bpTableSelect FileClient.receive` → `FileClient.write` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.receive` **and** `FileClient.write` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **Create a** `string literal` **and enter** `Selecting record(s) from db_employee table via table select...` **as the String value.**

   c. **Map the** `Selecting record(s) from db_employee table via table select...` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.input` **in the Input pane (right pane) of the Business Rule Designer.**

3 **Create the** `FileClient.write` → `otdSQLServer.DB_EMPLOYEESelectAll` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.write` **and** `otdSQLServer.DB_EMPLOYEESelectAll` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **Map** `text` **under** `FileClient.receive.Output` **in the Output pane of the Business Rule Designer, to the** `whereClause` **under** `otdSQLServer.DB_EMPLOYEESelectAll.Input` **in the Input pane.**

4 **Create the** `otdSQLServer.DB_EMPLOYEESelectAll` → `otdInputDTD_DBemployees.marshal` **rule.**

   a. **Add a Business Rule to the link between** `otdSQLServer.DB_EMPLOYEESelectAll` **and** `otdInputDTD_DBemployees.marshal` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **From the Business Rule Designer String menu, select** `String`**.**

      If String is not available from the String menu, click `Settings` on the String menu to open the Method Palette, and select `String` as an option from the String tab. This will add `String` to the String menu

   c. **Map** `EMP_NO` **under** `otdSQLServer.Select_psPSSelectAll.Output` → `output` → `DB_EMPLOYEESelectAllTableResponseTypeList` **in the Output pane of the Business Rule Designer, to the** `object1?` **input node of the** *String* **method box.**

   d. **Map the** `return string` **output node of the** *String* **method box, to** `EmpNo` **under** `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` **in the Input pane of the Business Rule Designer**

   e. **From the Business Rule Designer String menu, select** `String` **to add another** *String* **method box.**

   f. **Map** `RATE` **under** `otdSQLServer.Select_psPSSelectAll.Output` → `output` → `DB_EMPLOYEESelectAllTableResponseTypeList` **in the Output pane of the Business Rule Designer, to the** `object1?` **input node of the** *String* **method box.**

   g. **Map the** `return string` **output node of the** *String* **method box, to** `Rate` **under** `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` **in the Input pane of the Business Rule Designer**

   h. **Map nodes under** `otdSQLServer.Select_psPSSelectAll.Output` → `output` → `DB_EMPLOYEESelectAllTableResponseTypeList` **in the Output pane of the Business Rule**

Designer, to the corresponding nodes under `otdInputDTD_DBemployees.marshal.Input` → `DBemployees` → `X_sequence_A` **in the Input pane of the Business Rule Designer, as follows:**

- LAST_NAME → Lastname
- FIRST_NAME → Firstname
- LAST_UPDATE → LastDate

5 **Create the** `otdInputDTD_DBemployees.marshal` → `FileClient.write` **rule.**

a. **Add a Business Rule to the link between** `otdInputDTD_DBemployees.marshal` **and** `FileClient.write`**, and double-click the new Business Rule icon to open the Business Rule Designer.**

b. **Map** `text` **under** `otdInputDTD_DBemployees.marshal.Output` **in the Output pane of the Business Rule Designer, to** `text` **under** `FileClient.write.Input1` **in the Input pane.**

6 **Create the** `FileClient.write` → `FileClient.write` **rule.**

a. **Add a Business Rule to the link between** `FileClient.write` **element and** `FileClient.write`**, and double-click the new Business Rule icon to open the Business Rule Designer.**

b. **Create a** `string literal` **and enter** `TableSelect Done...` **as the String value.**

c. **Map the** `TableSelect Done...` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.input2` **in the Input pane (right pane) of the Business Rule Designer.**

Creating the bpUpdate Business Rules

The bpTableSelect business process describes how to select all records the SQL Server database using the Business Process Designer.

Note – The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.

▼ Create the bpUpdate Business Rules

1 **Double-click** `bpUpdate` **in the Projects window to open the Business Process Designer to the bpUpdate Business Process.**

2. **Create the** `bpUpdate FileClient.receive` → `FileClient.write` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.receive` **and** `FileClient.write` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **Create a** `string literal` **and enter** `Update the Rate and Last_update fields...` **as the String value.**

   c. **Map the** `Update the Rate and Last_update fields...` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.input` **in the Input pane (right pane) of the Business Rule Designer.**

3. **Create the** `FileClient.write` → `otdSQLServer.DB_EMPLOYEEUpdate` **rule.**

   a. **Add a Business Rule to the link between** `FileClient.write` **and** `otdSQLServer.DB_EMPLOYEEUpdate` **and double-click the new Business Rule icon to open the Business Rule Designer.**

   b. **Create a** `string literal` **and enter today's date as the String value.**

   c. **Map the** *today's date* **output node of the** *string literal* **method box, to** `LAST_UPDATE` **under** `otdSQLServer.DB_EMPLOYEEUpdate.Input` → `insert` **in the Input pane (right pane) of the Business Rule Designer.**

   d. **Create another** `string literal` **and enter** `888` **as the String value.**

   e. **Map the** `888` **output node of the** *string literal* **method box, to** `RATE` **under** `otdSQLServer.DB_EMPLOYEEUpdate.Input` → `insert` **in the Input pane (right pane) of the Business Rule Designer.**

   f. **Map the** `text` **under** `FileClient.receive.Output1` **in the Output pane of the Business Rule Designer, to** `whereClause` **under** `otdSQLServer.DB_EMPLOYEEUpdate.Input` → `insert` **in the Input pane (right pane) of the Business Rule Designer.**

4. **Create the** `otdSQLServer.DB_EMPLOYEEUpdate` → `FileClient.write` **rule.**

   a. **Create a** `string literal` **and enter** `Update Done` **as the String value.**

   b. **Map the** `Update Done` **output node of the** *string literal* **method box, to** `text` **under** `FileClient.write.Input1` **in the Input pane of the Business Rule Designer.**

5. **Save your Project.**

Next Steps For your next step, see "Creating the Connectivity Maps " on page 45.