OpenNode2

Plugin Development
using Altova MapForce®

Version: 1.0

Revision Date: 6/26/2009

Prepared By:

WINDSOR
SOLUTIONS, INC.

4000 Kruse Way Place, 2-285
Lake Oswego, OR 97035
503-675-7833

Environmental Information

e×change
Network

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Revision History

| Version	Author	Date
1.0	Windsor	June, 2009

THIS PAGE INTENTIONALLY LEFT BLANK

1   Introduction

This guide provides a tutorial demonstrating how to develop a plugin for the Exchange Network OpenNode2 using Altova MapForce®. After completing this tutorial, a developer will have the skills and knowledge to build custom plugins that either publish or consume data over the Exchange Network using either the .NET or Java implementation of OpenNode2.

In the context of this project, MapForce is used to map relational databases and other data sources to XML or vice versa using the graphical mapping interface. MapForce-generated code is then compiled, integrated into a pre-built OpenNode2 plugin wrapper, and uploaded to OpenNode2.

The following sections describe inbound and outbound data exchanges. MapForce can be used to develop components for both types of exchanges, as described in the sections below.

1.1   Outbound Data Exchange Example

The diagram below illustrates the most common architecture for an outbound OpenNode2 data exchange:
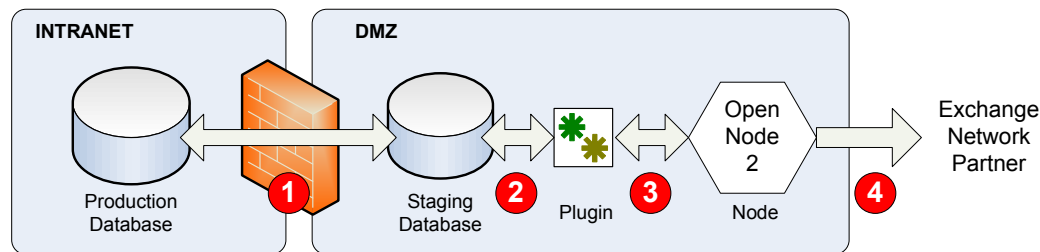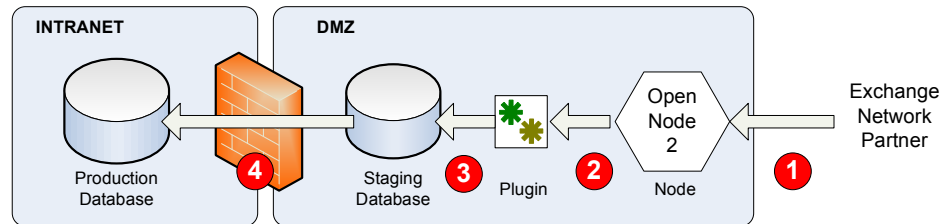

The list below describes each step in the diagram:

1. On a periodic basis, data from an agency's production database is replicated to a separate staging database. Typically, the staging database architecture is similar in design to the associated XML schema. This facilitates faster and more efficient mapping from the database to XML for servicing on-demand data requests.

2. An event triggers the node to retrieve data from the staging database. This could be triggered by a query from an external partner or the execution of a scheduled job from within the node.

3. The plugin reads data from the staging database, transforms it to XML, and returns the XML file to the node.

4. The node performs some action with the XML file, such as returning it to the data requestor or submitting it to a predefined network partner. In conjunction with this activity, OpenNode2 can be configured to send notifications to one or more email recipients.

In the scenario above, MapForce can be used to assist in developing the data transformation between the production and staging databases (step 1) and generating the XML file from the data in the staging database (step 3).

1.2 Inbound Data Exchange Example

An inbound data exchange accepts data from an external network partner and puts it to use within the agency. The following diagram illustrates a common architecture for an inbound data exchange:

The list below describes each step in the diagram:

1. A file is submitted by an external Network partner to OpenNode2.

2. On a timed interval, OpenNode2 inspects the inbound file and routes it for processing by the appropriate plugin (based on the data flow name specified in the submission).

3. The plugin transforms the XML into a set of relational database tables, inserting, updating or deleting data as appropriate.

4. A separate process reads the data from the staging tables and writes them to the agency's production database.

Similar to the outbound data exchange scenario, MapForce can be used to assist in developing the data transformation from XML to the staging database (step 3) and transforming data between the staging and production databases (step 4).

1.3 Limitations of MapForce-generated Plugins

There are some important limitations to developing plugins using MapForce.

- Each MapForce-generated plugin can only offer a single outbound data service (Query or Solicit) OR inbound submission processor (Submit).

- While Query/Solicit parameters are supported (zero or more), they are limited to the string data type.

- It is not possible to pass an array of values to an input parameter.

- MapForce-generated plugins may offer slower performance than a custom-built plugin. Overall performance is generally very good but performance can be degraded by many complex transformations.

- Schedules created in OpenNode2 must pass parameters in By Index, not By Name.

2   Tutorials

In this tutorial, you will build a plugin for the fictitious BEACHDEMO data exchange. The plugin will offer a single data service "GetBeachByBeachNameOrWaterbodyName_v2.1". As the name suggests, the service will enable a Network partner to query the node for beach data. The service will provide the requester to search by Beach Name (i.e. "Traverse City Beach") and/or Waterbody Name (i.e. "Lake Michigan").

When this service in invoked, the plugin will read data from a SQL Server or Oracle database using the user-provided criteria, transform the results to an XML file conforming to the Beach Notification v2.1 schema, and return the data to the requester.

Prerequisites

The following prerequisites must be met in order to complete this tutorial

- Altova MapForce 2009

- Visual Studio .NET 2005, 2008, or Java Development kit (JDK) 1.5 and Eclipse 3.3 or later

- Oracle or Microsoft SQL Server database

- Installed and operational version of either the .NET or Java OpenNode2 node

- Installed copy of NodeClientLite2 in order to test the exchange.

2.1   Tutorial Setup

2.1.1   Unpacking the Tutorial Files

This tutorial is provided in the form of a ZIP file containing all materials needed to complete the exercises in this document. It is assumed that the ZIP file will be extracted to **C:\MapForceTutorial**.

2.1.2   Setting up the Database Environment

This tutorial requires that either a SQL Server or Oracle database be created and used as a data source for the plugin. Follow the steps below to establish the test database.

SQL Server

1. On a SQL Server instance, create a new database named **BEACHDEMO**.

2. Connect to the **BEACHDEMO** database using SQL Server Management Studio Express or similar tool.

3. Run the **Create_DB_and_Tables (SQL Server).sql** script located in the **C:\MapForceTutorial\SQL** folder. Two tables will be created; BEACH and BEACH_ACTIVITY.

4. Next, run the **Insert_Sample_Data (SQL Server).sql** to insert two records into each table. The script is also located in the **C:\MapForceTutorial\SQL** folder.

Oracle

1. On an Oracle database instance, create a new schema named **BEACHDEMO**.

2. Connect to the **BEACHDEMO** database using a tool such as Oracle SQL Developer.

3. Run the **Create_DB_and_Tables (Oracle).sql** script located in the **C:\MapForceTutorial\SQL** folder. Two tables will be created; BEACH and BEACH_ACTIVITY.

4. Next, run the **Insert_Sample_Data (Oracle).sql** to insert two records into each table. The script is also located in the **C:\MapForceTutorial\SQL** folder.

## 2.2 Developing a Outbound Data Exchange

In this tutorial, you will:

1. use MapForce, map the **BEACHDEMO** database to the Beach Notification schema and generate the code in either C# or Java,

2. compile the code into either a .NET assembly or Java JAR file,

3. integrate the **MapForceBridge** plugin adapter into either a ZIP file (for .NET) or JAR file (for Java),

4. upload and configure the plugin to OpenNode2,

5. and test the data service using a Node Client.

### 2.2.1   Map the Database to XML using MapForce

> **Note**: This tutorial does not attempt to cover the nuances of using MapForce. Please see the tutorials that are available in the MapForce Help file or on the Altova website for more information on using MapForce.

Add the Database and XML Schema to the Design Surface

1. Open MapForce. By default, a new, blank Design will be displayed.

2. On the **Insert** menu, choose **Database**. Choose either **SQL Server (ADO)** or **Oracle (ODBC)**, depending on the database type you are using.

3. In the Insert Database Objects dialog, choose the BEACH and BEACH_ACTIVITY tables from the appropriate database/schema.
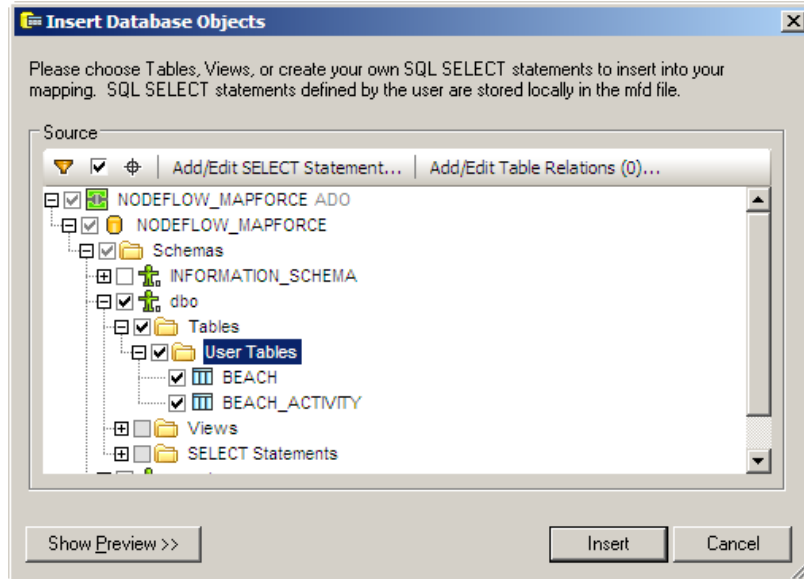
4. After completing the Connection Wizard, the designer will now display the BEACH and BEACH_ACTIVITY tables on the design surface.
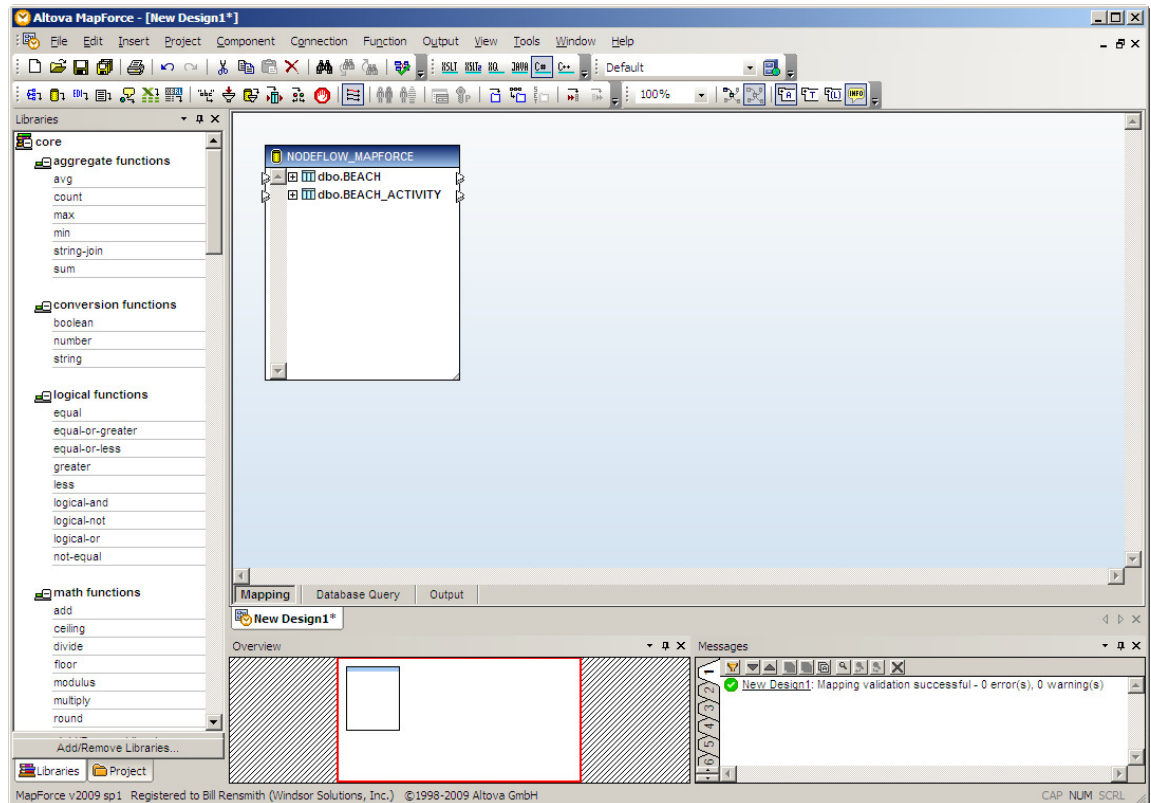
5. On the Insert Menu, choose **XML Schema/File…** Navigate to the **C:\MapForceTutorial\BeachNotification_Schema_v2.1\2\1** directory and choose the **BEACHES_Notification_v2.1.xsd** schema file.

6. After clicking **OK**, a dialog box will display, asking if you wish to supply a sample XML file. Click **Browse** and navigate to the **C:\MapForceTutorial\BeachNotification_Schema_v2.1** directory and choose the **Beach_ExampleXMLFile_v2.1.xml** file. Click **OK.**

7. After adding the XML schema and sample instance file, the designer will add the XML schema on the design surface.
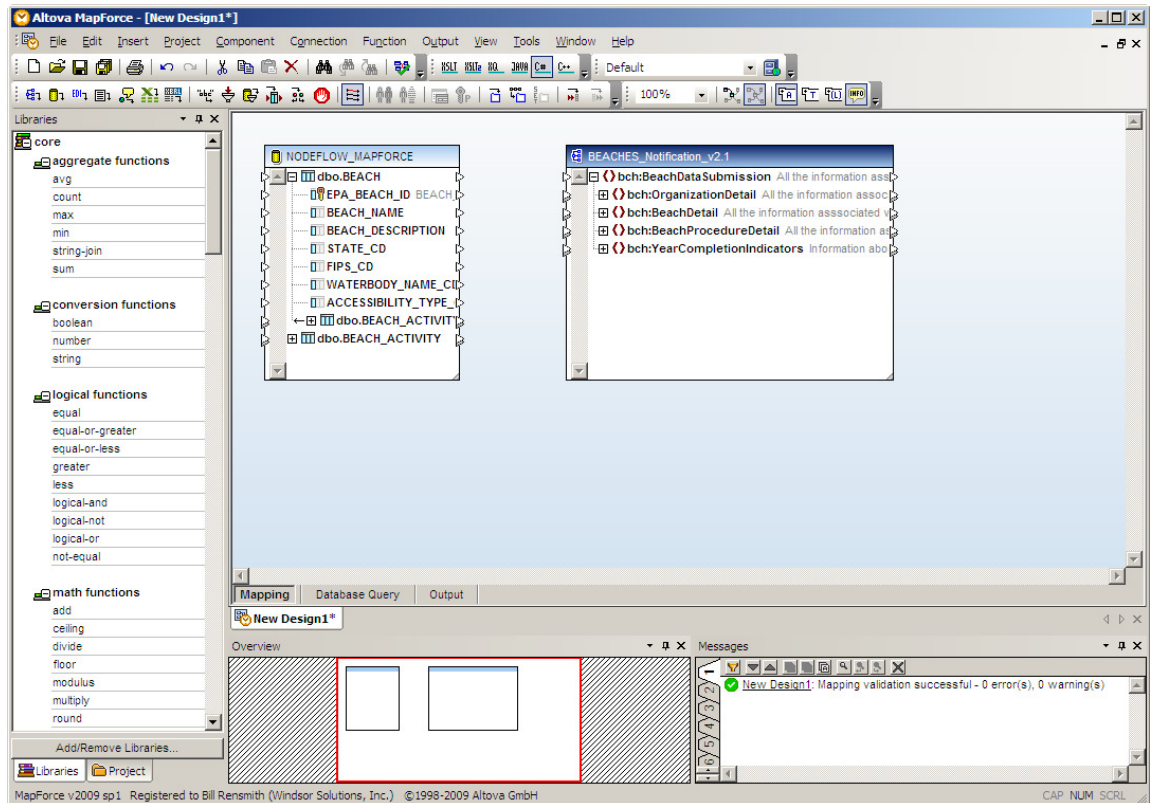
Map Database Tables and Fields to XML Elements

8. Next, map the database tables to the appropriate XML schema elements as follows:

    a. Drag a line between the BEACH table and the BeachDetail XML schema node. This indicates that there is a 1-to-1 relationship between records in the BEACH table and the BeachDetail schema block.

    b. Expand the BeachDetail schema node and the BeachNameDetail sub-node in the XML schema pane.

    c. Drag a line between EPA_BEACH_ID and BeachIdentifier.

    d. Drag a line between BEACH_NAME and ProgramInterestName.

    e. Drag a line between BEACH_DESCRIPTION and ProgramInterestDescription.

    f. Drag a line between STATE_CD and ProgramInterestStateCode.

    g. Drag a line between FIPS_CD and ProgramInterestFIPSCountyCode.

    h. Drag a line between WATERBODY_NAME_CD and WaterBodyNameCode.

    i. Expand the BeachAccessibilityDetail Node in the XML schema component.

j.   Drag a line between ACCESSIBILITY_TYPE_CD and BeachAccessibilityType.

9. Next, map the list of beach advisories (i.e. "activities") from the database to the XML schema elements as follows:

a.   Collapse the BeachNameDetail node and expand the BeachActivityDetail node in the XML schema component.

b.   Expand the BEACH_ACTIVITY table in the database component. Be sure that you are working with the BEACH_ACTIVITY table nested beneath the BEACH table, not the BEACH_ACTIVITY table that is a sibling to the BEACH table.

c.   Drag a line between the BEACH_ACTIVITY table in the database pane and the BeachActivityDetail node in the XML schema pane. This indicates that there is a 1-to-1 relationship between records in the BEACH_ACTIVITY table and the BeachActivityDetail schema block.

d.   Drag a line between ACTIVITY_TYPE and ActivityTypeCode.

e.   Drag a line between ACTIVITY_NAME and ActivityName.

f.   Drag a line between START_DATE and ActivityActualStartDate.

g.   Drag a line between REOPEN_DATE and ActivityActualStopDate.

h.   Drag a line between REASON_TYPE_CD and the ActivityReasonType element located within the ActivityReasonDetail node.

i.   Drag a line between POLLUTION_SOURCE_CD and the ActivtySourceType element located within the ActivitySourceDetail node.

j.   Drag a line between INDICATOR_TYPE_CD and the ActivityIndicatorType element located within the ActivityIndicatorDetail node.

k.   Your final mapping diagram should appear similar to the image below:
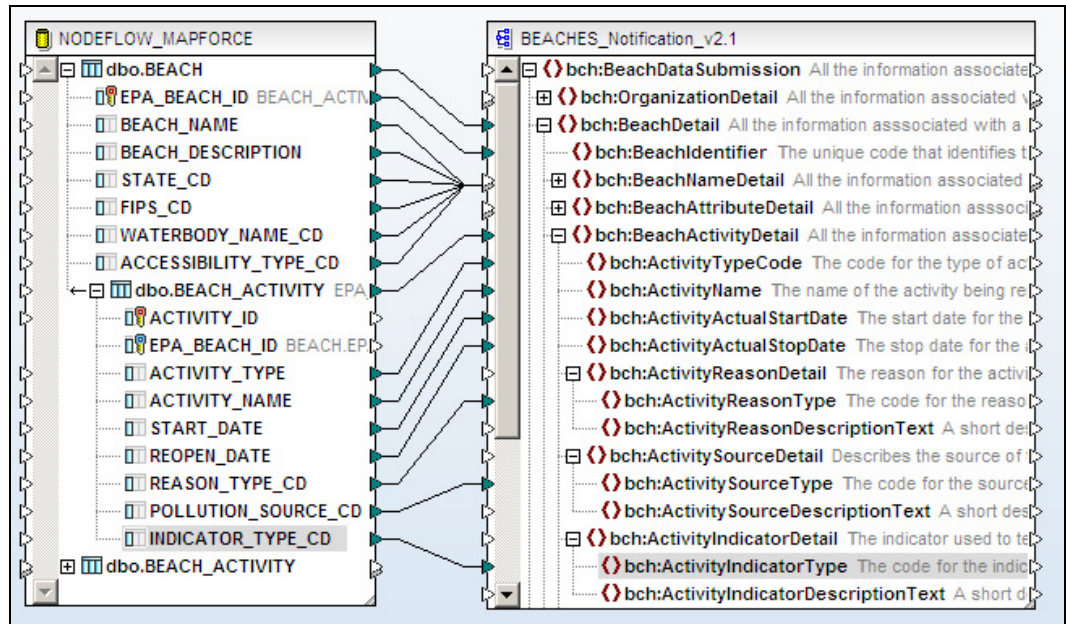
10. Click the Output tab at the bottom of the screen to preview your transformation.

11. Additionally, you may click the Validate Output toolbar button ⬚ to validate the XML file against the schema. If the mapping is done properly, the file should validate.

12. Save the mapping file as **DBtoXML.mfd** to **C:\MapForceTutorial\DBtoXML**.

2.2.2   Define Input Parameters

Next, you will define the input parameters used to filter the output based on user input. These parameters will be exposed through the node, allowing an external partner to search based on Beach Name and/or Waterbody Name. The input parameters will be constructed to meet the following requirements:

- The service will search on any part of the waterbody or beach name (wildcard search) without requiring the user to explicitly provide the wildcard characters.

- The service will allow the user to search by either the beach name, waterbody name, or both.

- The service will return all records if no input criteria are provided.

Create the SQL WHERE Condition to Filter Results

1. Return to the Mapping tab.

2. In the **Insert** menu, choose **SQL-WHERE Condition**.

3. Drag a line from the BEACH table to the table connector on the **SQL-WHERE** component as shown below:
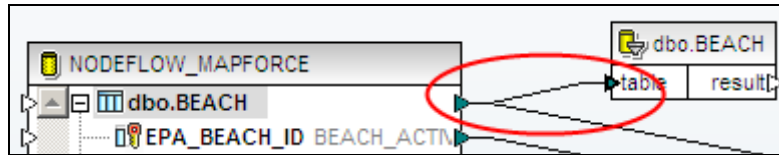
4. The SQL-WHERE Properties dialog box will display. Type the following statement in the text area within the dialog box:

    ```
    BEACH_NAME LIKE :BeachName AND WATERBODY_NAME_CD LIKE
    :WaterbodyName
    ```

    Note: The colon prefix indicates that the item is an input parameter. See the MapForce documentation for more information.

5. In the Parameters table at the bottom of the dialog, you may wish to change the type of each parameter to **string** from **(auto-detect by database)**. This step may not be necessary depending on the database platform.

6. Click **OK** to close the dialog box. The two input parameters will now display in the **SQL-WHERE** component as show below:
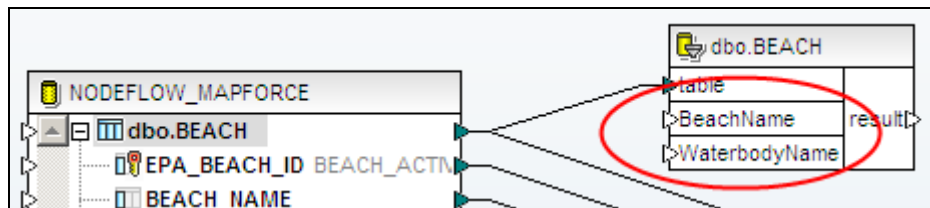
Create Input Parameters

7. Next you will need to create the two input parameters and link them to the **SQL-WHERE** parameters. Right-click on any blank area within the design surface and choose **Insert Input…** The Create Input dialog will display as show below:

8. Type the name **BeachName**, check the **Input is Required** checkbox (if needed), and click **OK**.

> **Note:** If the **Input Is Required** checkbox is left unchecked, the query will return no data when the query is executed with one or more blank parameters.

9. Repeat the previous two steps to create a second input named **WaterbodyName**.

Add Wildcard Characters to the Input Criteria

Next, you will need to concatenate the wildcard characters (%) to the beginning and end of the input parameters to allow the user to search on any part of the beach name or waterbody name.

10. Right-click on a blank area of the design service and choose **Insert Constant**…

11. When the dialog appears, enter a percent sign (**%**) and click OK.

12. Repeat the previous two steps to create a second, identical constant.

13. Now drag a **concat** item from the **string functions** library in the left-hand pane onto the design surface.

14. Link the two constants and the **BeachName** input parameter to the **concat** component as illustrated below:
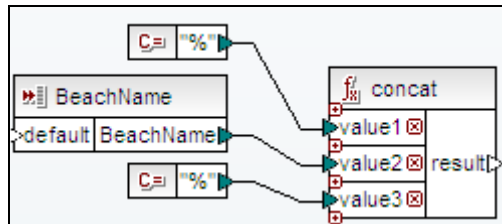

15. Next, drag a line between the result of the **concat** component and the **BeachName** parameter on the **SQL-WHERE** component. The design should appear similar to the image below:
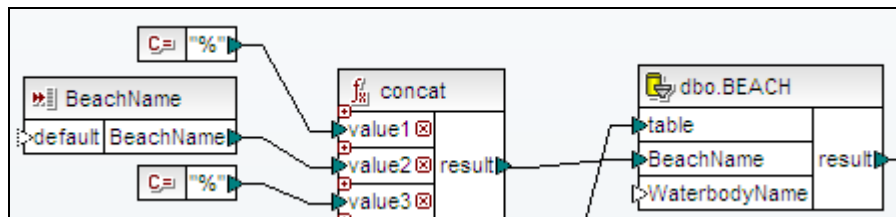
16. Repeat steps 10 through 15 above for the **WaterbodyName** input. The design should now appear similar to the image below:

---
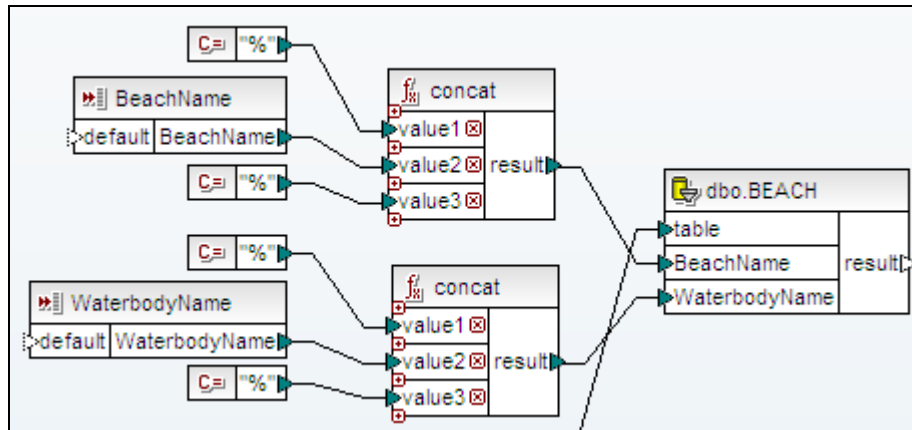
17. Lastly, drag a line between the result on the **SQL-WHERE** component and the BeachDetail node in the XML schema pane. The following message box will appear:
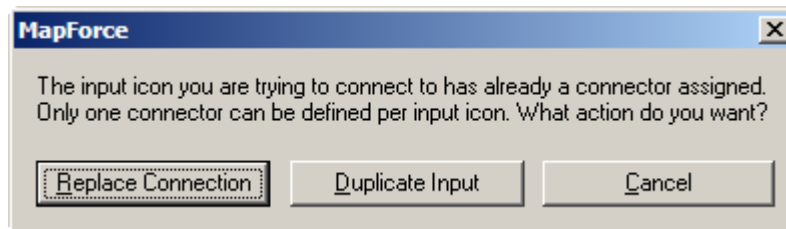
18. Choose **Replace Connection**. This will remove the line that previously connected the BEACH table directly to the BeachDetail XML node with a new line from the result on the **SQL-WHERE** component to the BeachDetail XML node.

The design should appear similar to the image below:
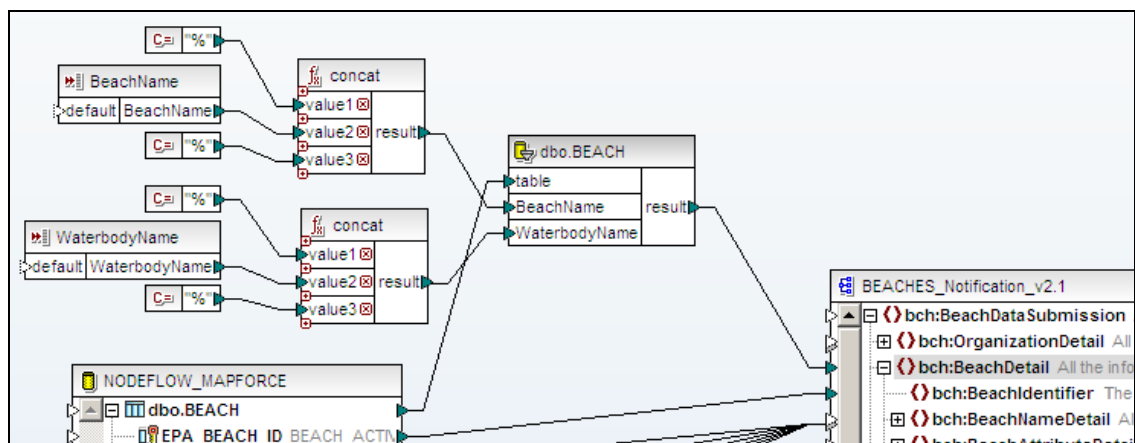
The input parameters are now configured properly.

19. Lastly, save the mapping.

Note: If you wish to create a data service that takes an input parameter without performing a wildcard search, you can link the input parameter directly to the **SQL-**

WHERE component. If this is done, you must leave the **Input is Required** checkbox unchecked. If it is left checked and a user does not supply a value when executing the query, a node exception will occur.

2.2.3   Generate Code and Assemble the Plugin

Please see the appropriate section below for either .NET or Java.

2.2.3.1   Microsoft .NET

1. Determine whether you will be generating code for Visual Studio 2005 or 2008. By default, MapForce will generate a Visual Studio 2008 project. To modify this setting, select the **Tools** menu, choose **Options**, select the **Generation** tab, and choose the appropriate code generation setting.

2. From the **File** menu, choose **Generate Code in -> C# (sharp)**.

3. When prompted, choose the **C:\MapForceTutorial\DBtoXML** folder and click **OK**. The Visual Studio solution will be generated in the selected folder.

4. Open the Mapping.sln located in **C:\MapForceTutorial\DBtoXML\Mapping** using Visual Studio 2005 or 2008. Next, compile the project by selecting **Build Solution** from the **Build** menu.

> Note that MapForce generates a sample console project named **Mapping**. While not necessary for the purposes of this tutorial, it is interesting to examine the code in **MappingConsole.cs** to see how the transformation is invoked and how it can be tested from the command line.

5. Navigate to the **C:\MapForceTutorial\DBtoXML\Mapping\bin\Debug** folder to view the compiled output.
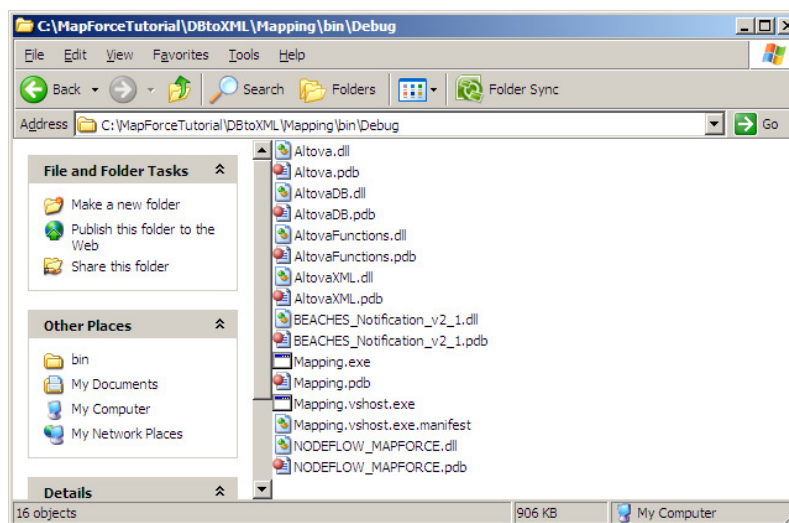
6. Visit the OpenNode2 code repository and download the compiled .NET MapForceBridge Plugin.

7. Extract the downloaded plugin files to a directory on your computer. Copy the contents of the folder to the clipboard.

8. Navigate back to **C:\MapForceTutorial\DBtoXML\Mapping\bin\Debug** and paste the plugin files into the directory alongside the compiled MapForce output. The MapForce Bridge plugin files will be added to the directory as shown in the image below:
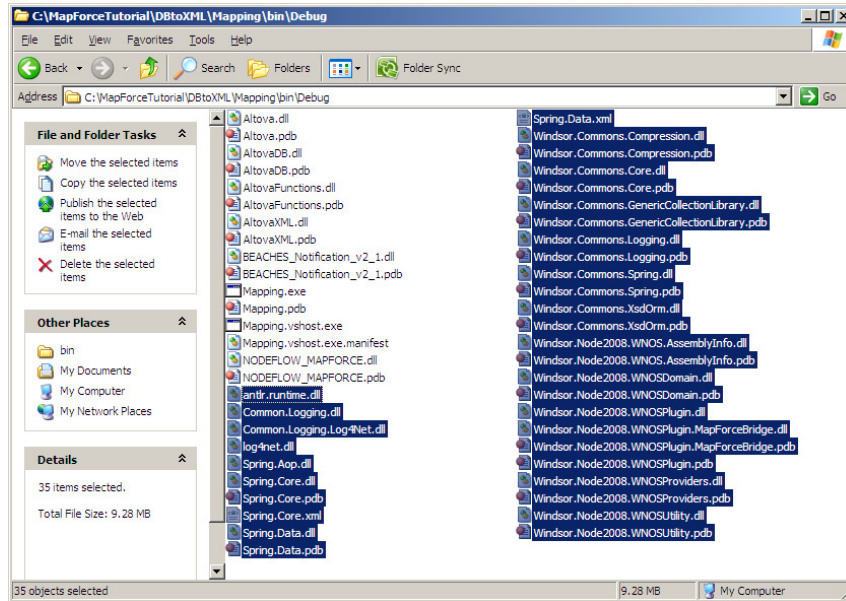
9. Press **CTRL+A** to select all the files, right-click on any file and choose **Send to -> Compressed (zipped) Folder**.

10. Rename the newly-created ZIP file **MapForceTutorial.zip**. (optional step)

You have successfully created the plugin and are now ready to install it to OpenNode2. Please skip to section 2.2.4 to continue.

2.2.3.2   Java

1. From the **File** menu, choose **Generate Code in -> Java**.

2. When prompted, choose the **C:\MapForceTutorial\DBtoXML** folder and click **OK**. The Java source code will be generated in the selected folder.

3. Compile the Java plugin using the generated ANT build script. The final output should be a single **Mapping.jar** file.

4. Visit the OpenNode2 code repository and download the compiled Java MapForceBridge Plugin (**mapforcebridge.zip**). This zip file contains a single entry, mapforcebridge.jar.

5. Add **Mapping.jar** to mapforcebridge.zip.

6. Name the zip file **MapForceTutorial.zip**. (optional step)

You have successfully created the plugin and are now ready to install it to OpenNode2.
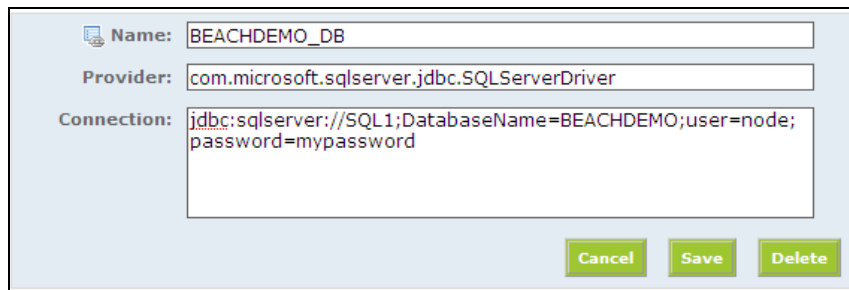
2.2.4   Install and Configure the Plugin

Now that the plugin is built, it can now be installed on OpenNode2.

1. Log into the OpenNode2 Admin Utility.

2. <u>**Java Node only:**</u> The Java node requires that a data source be established for the plugin to connect to the BEACHDEMO database/schema.
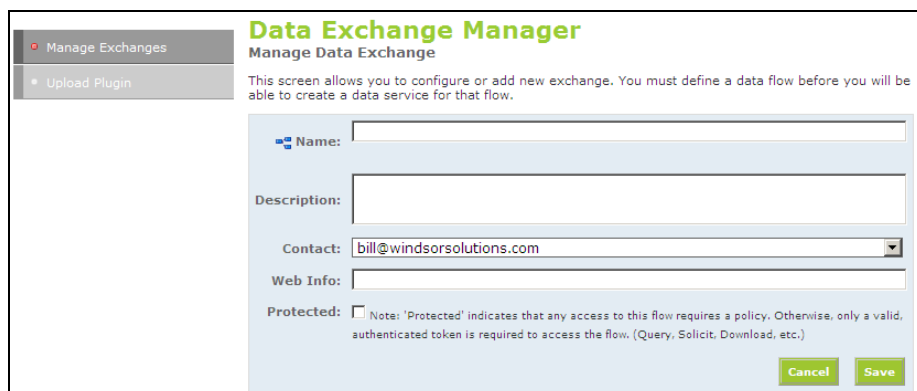
    To do this, click the **Configuration** tab. Click **Data Sources** on the left-hand navigation menu. Click **Add** to create a new data source. Name the connection **BEACHDEMO_DB**. Set the appropriate connection settings for the BEACHDEMO database/schema. See the OpenNode2 Administrator's Guide for more information.

    The image below displays an example of a SQL Server connection:

Add the BEACHDEMO Exchange

3. Click the **Exchange** tab and click the **Add Exchange** button. The following screen will be displayed:

4. In the **Name** field, type **BEACHDEMO**.

5. If necessary, set yourself as the contact in the **Contact** drop down box.

6. In the **URL** field, enter
   http://www.exchangenetwork.net/exchanges/water/beach_notif.htm.

7. Click **Save** to create the new exchange. You will now see BEACHDEMO display in the list of installed exchanges.

Upload the Plugin

8. Next, click **Upload Plugin** from the left-side navigation menu.

9. Click the **Browse…** button and select the **MapForceTutorial.zip** file created in the previous section.

10. Choose **BEACHDEMO** from the **Exchange** drop down box.

11. Click **Upload** to complete uploading the plugin.

Add the Query Service

Next, a service will be added that will allow external partners to query the node for a list of beaches by beach name or waterbody name.

12. On the Data Exchange Manager tab, click **Add Service**, located beneath the BEACHDEMO exchange header.
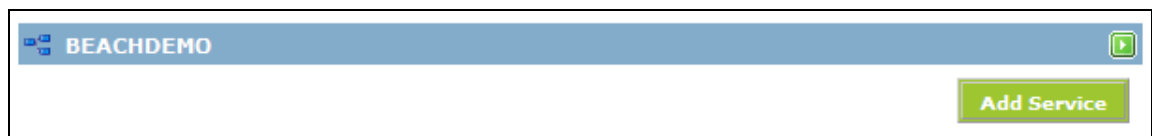
13. In the **Service** field, enter **GetBeachByBeachNameOrWaterbodyName_v2.1** for the service name.

14. Select the **MapForceBridge** item from the **Implementer** drop down box. (This should be the only choice available)

    a. **Java Node only**: Click the **Next** button

15. Select **Query Or Solicit** from the **Type** drop down box. This will expose the service as both an Exchange Network Query and Solicit primitive.

16. Perform the following step for your specific environment:

a. **.NET Node only:** Type the connection string to the database in the **Database Connection String** field. This will be passed to the MapForce mapping code to establish a connection to the source database.

> **Note:** For .NET users, the connection string can be found in the MapForce-generated **MappingConsole.cs** file.

b. **Java Node only:** In the Source **Data Provider** drop down box, select the BEACHDEMO_DB connection created at the beginning of this section.

17. Click **Save** to save the new service.

2.2.5   Test the Service

Lastly, you will invoke the new service. OpenNode2 will return the beaches that match the search in the form of the Beach Notification v2.1 schema conformant XML file.

1. Launch NodeClientLite2.

2. Connect to your node from the **Connect to Node** screen by either typing in your node URL or selecting the node from the Connection History list. Click **Connect** to initiate the connection.

Register the new Exchange and Service with NodeClientLite2

Next, you will need to register the service with NodeClientLite2. Until you do this, NodeClientLite2 will not know what parameters the service will accept.

3. Click **Services** from the **My Preferences** navigation menu. The Data Services screen will appear:
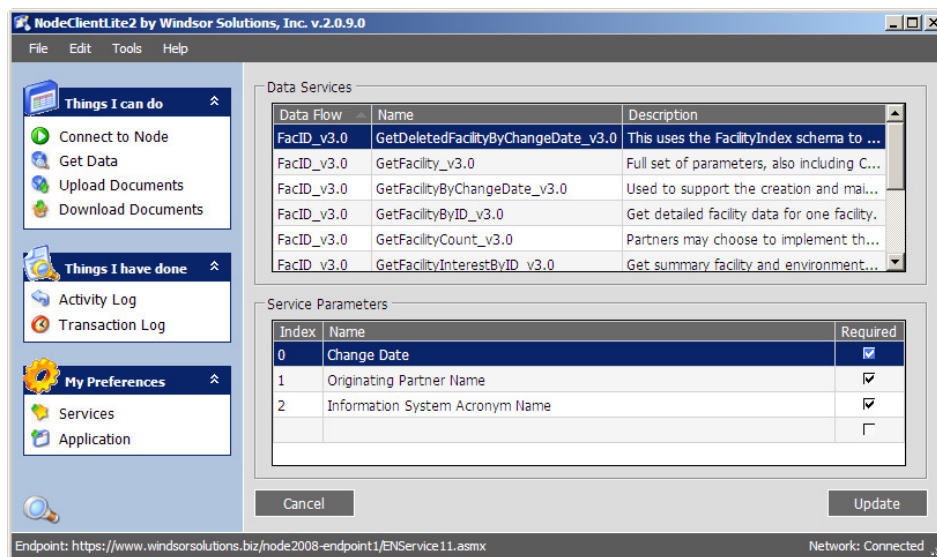
4. Scroll to the bottom of the Data Services list to the last blank row. Click into the Data Flow field and type **BEACHDEMO**. Click into the adjacent field and type **GetBeachByBeachNameOrWaterbodyName_v2.1**. Click off the row to save. The list will automatically resort and the BEACHDEMO service will appear at the top of the list

5. Select the **BEACHDEMO** service in the top list and add the two Service Parameters to the bottom grid as displayed in the image below:


Be sure to leave the **Required** checkboxes unchecked.

Testing the "GetBeachByBeachNameOrWaterBodyName" Query Service

6. Click **Get Data** from the **Things I can do** navigation menu. The Get Data screen will appear:

7. In the **Data Flow** field, type or select **BEACHDEMO**

8. In the **Service** field, select **GetBeachByBeachNameOrWaterbodyName_v2.1**. The screen will refresh to display the two optional input parameters

9. To return all records, leave both parameters blank

10. Click **Submit**.

11. One the query is done executing, NodeClientLite2 will display a confirmation screen that the query was executed successfully.
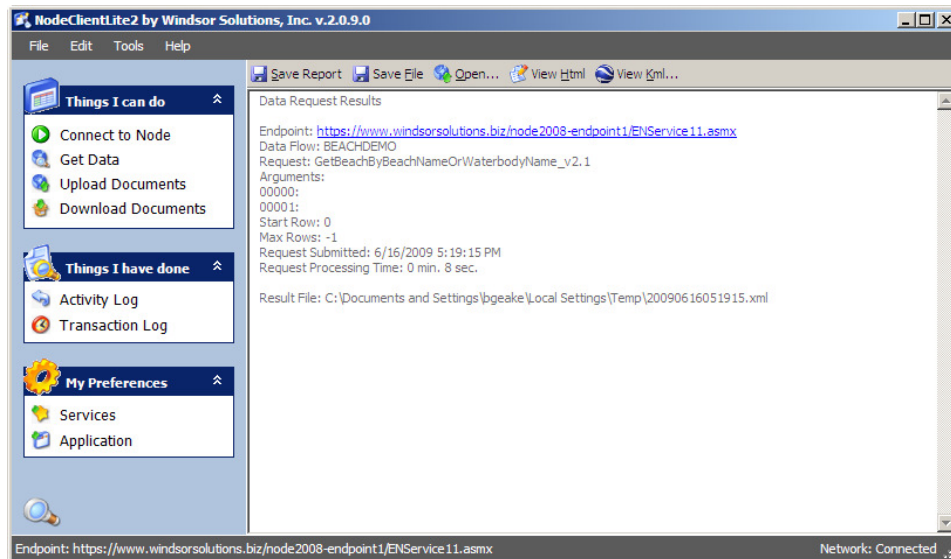
12. Copy the result file path from the window and paste into the Run prompt on the Windows Start menu. The file will display in Internet Explorer by default. Note that all records were returned since no input criterion was provided.
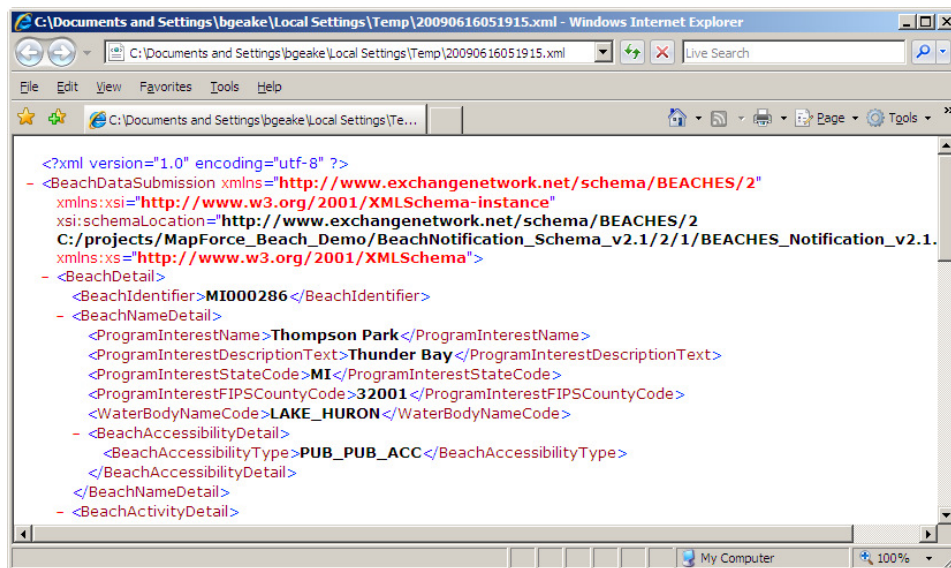
13. Repeat steps 6 through 12 only change the criteria to provide a beach name of **Traverse** and no waterbody name. Note that only the data for Traverse City Beach is returned.

14. Repeat steps 6 through 12 only change the criteria to provide a waterbody name of **huron** and no beach name. Note that only the data for Thompson Park is returned.

2.3   Developing an Inbound Data Exchange

In this tutorial, you will:

- Use MapForce to map the Beach Notification schema to the **BEACHDEMO** database and generate the code in either C# or Java,

- compile the code into either a .NET assembly or Java JAR file,

- integrate the **MapForceBridge** plugin adapter into either a ZIP file (for .NET) or JAR file (for Java),

- upload and configure the plugin to OpenNode2,

- and perform a test submission using a Node Client.

This tutorial includes a sample XML file to be submitted to the node for the purposes of updating the BEACHDEMO database. It is important to be familiar with the data contained in the file to understand how the database will be updated. The file contains:

- A replica of the existing Thompson Park record. The BEACH record will be **updated** with this data.

  o A new advisory for Thompson Park starting 6/10/2009. This data will be **inserted** into the BEACH_ACTIVITY table.

- A new beach named My New Beach. This data will be **inserted** into the BEACH table.

  o An advisory for My New Beach. This data will be **inserted** into the BEACH_ACTIVITY table.

2.3.1   Map the XML to the Database to using MapForce

Add the XML schema and Database to the Design Surface

1. Follow steps 1-7 in Section 2.2 to add the XML schema and database tables to the MapForce design surface. When complete, the design surface should appear similar to the image below:
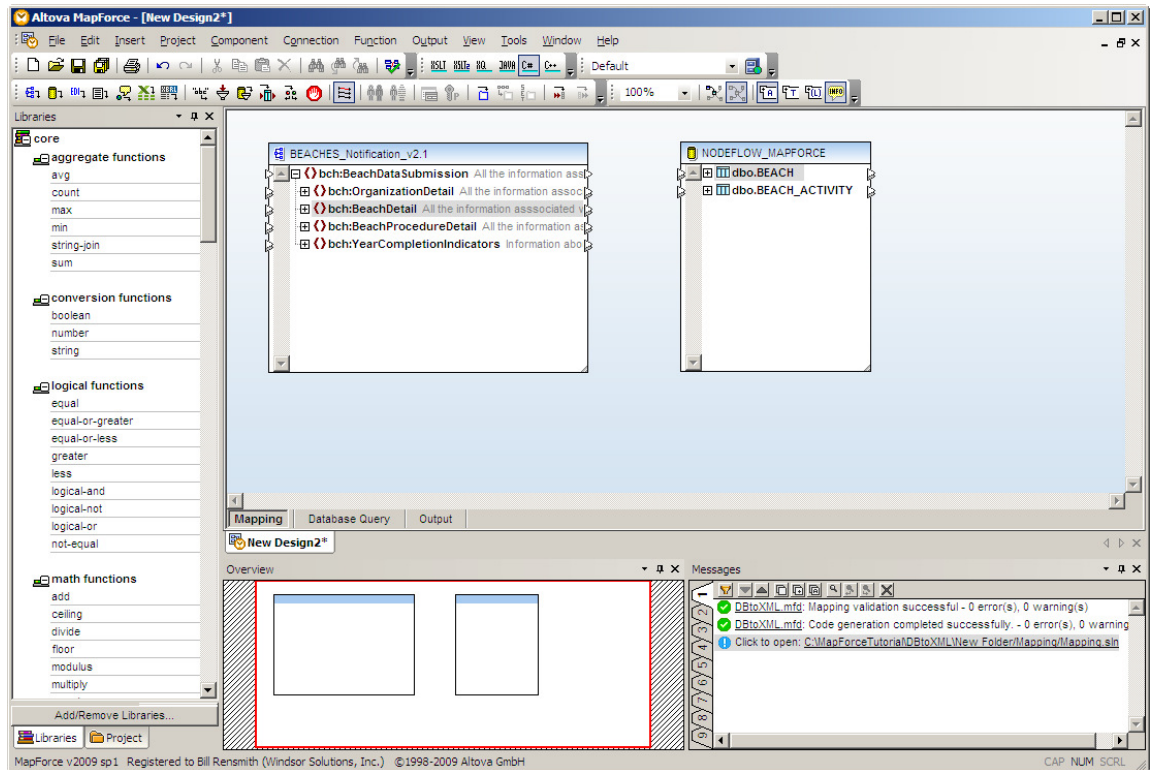
Map XML Schema Elements to Database Tables and Fields

2. Next, map the XML schema to database tables and field by dragging connector lines from the XML schema component to the database component as follows:

     a. Drag a line from the BeachDetail element to the BEACH table.

     b. Expand the BeachDetail schema block and the BEACH table.

     c. Drag a line from BeachIdentifier to EPA_BEACH_ID.

     d. Expand the BeachNameDetail schema block.

     e. Drag a line between ProgramInterestName and BEACH_NAME.

     f. Drag a line between ProgramInterestDescriptionText and BEACH_DESCRIPTION.

     g. Drag a line between ProgramInterestStateCode and STATE_CD.

     h. Drag a line between ProgramInterestFIPSCountyCode and FIPS_CD.

     i. Drag a line between WaterbodyNameCode and WATERBODY_NAME_CD.

     j. Expand the BeachAccessibilityDetail schema block.

k. Drag a line between BeachAccessibilityType and ACCESSIBILITY_TYPE_CD.

3. Next, map the list of beach advisories (i.e. "activities") from the XML schema to the database fields as follows:

a. Collapse the BeachNameDetail node and expand the BeachActivityDetail node in the XML schema component.

b. Expand the BEACH_ACTIVITY node in the database component. Be sure that you are working with the BEACH_ACTIVITY node nested beneath the BEACH node, not the BEACH_ACTIVITY node that is a sibling to the BEACH node.

c. Drag a line between the BeachActivityDetail node in the XML schema component and the BEACH_ACTIVITY table in the database component. This indicates that there is a 1-to-1 relationship between records in the BeachActivityDetail schema block and the BEACH_ACTIVITY table.

d. Drag a line between ActivityTypeCode and ACTIVITY_TYPE.

e. Drag a line between ActivityName and ACTIVITY_NAME.

f. Drag a line between ActivityActualStartDate and START_DATE.

g. Drag a line between ActivityActualStopDate and REOPEN_DATE.

h. Drag a line between the ActivityReasonType element located within the ActivityReasonDetail node and the REASON_TYPE_CD field.

i. Drag a line between the ActivtySourceType element located within the ActivitySourceDetail node and the POLLUTION_SOURCE_CD field.

j. Drag a line between the ActivityIndicatorType element located within the ActivityIndicatorDetail node and the INDICATOR_TYPE_CD field.

Add a Generated GUID for the BEACH_ACTIVITY.ACTIVITY_ID

The target database requires a primary key for the BEACH_ACTIVITY table in the form of a GUID. Since the source XML file does not supply a unique identifier, you must instruct MapForce to generate it automatically.

4. Drag a **create-guid** function from the generator functions category of the functions window onto the designer

5. Drag a line connecting the **create-guid** component to the ACTIVITY_ID field.

Your final mapping diagram should appear similar to the image below:
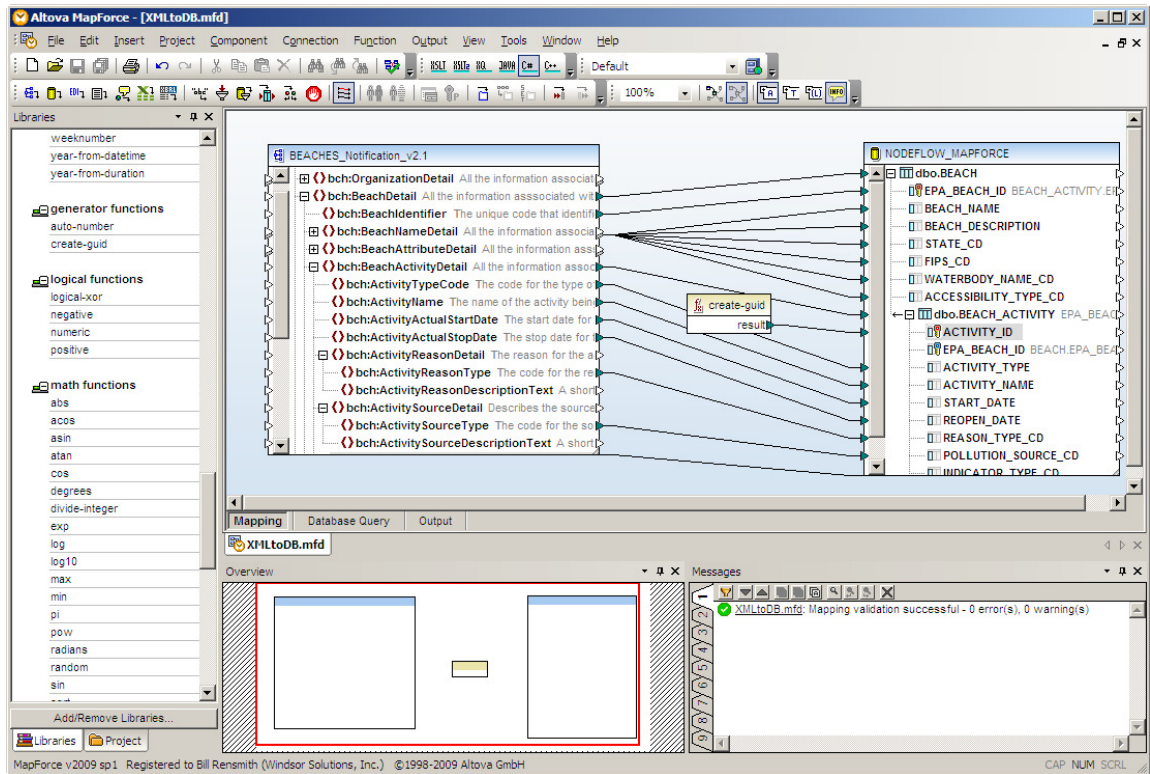
Set INSERT and UPDATE conditions

MapForce can determine whether to insert, update or delete records based on the rules in the mapping. In this example, the following business rules will be implemented:

- If the BeachIdentifier in an XML file matches an existing EPA_BEACH_ID in the database, update the record with new or changed data from the XML file.

- If the BeachIdentifier in an XML file is not found in the database, INSERT the new beach.

- If the XML file contains beach advisories that already exist in the database (based on a matching BeachIdentifier and StartDate), update the record.

- Otherwise, insert the remaining advisories.

Follow the steps below to set the INSERT and UPDATE conditions.

6. Right-click on the BEACH table in the database component and select **Database Table Actions**. The Database Table Actions dialog box will appear:
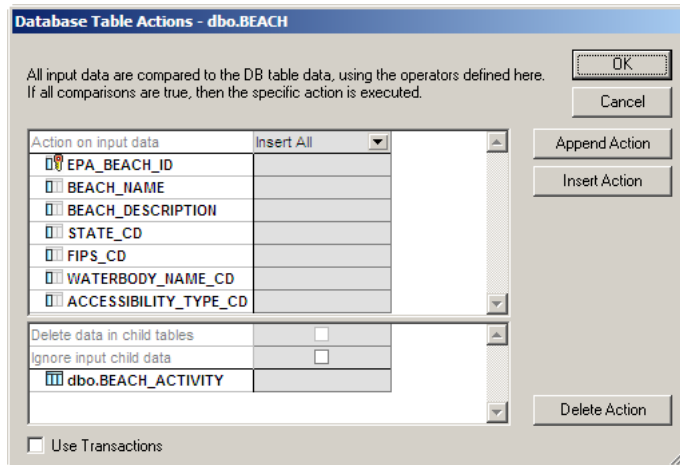
7. Click the **Insert Action** button.

8. Select **Update If…** from the drop down box from the left-most column.

    Note that the heading of the right-most column now reads **Insert Rest**. This indicates that MapForce will attempt to insert any records that do not match the conditions specified in columns to the left.

9. Select **equal** in the **EPA_BEACH_ID** row in the **Update If** column. This tells MapForce to update the database record if an existing record is found in the XML input file based on the mapping to the BeachIdentifer element in the designer.

10. Uncheck the **Delete data in child tables checkbox.**

    > **Note:** If we left this option checked, all child advisory data would be deleted when an input XML file contained a beach that existed in the database. This option would be appropriate if the XML file was a "full replace" of the database data. In the case of an "incremental add" scenario such as this example, the delete option should be turned off so historical child data is not deleted.

11. The dialog should now appear similar to the image below:
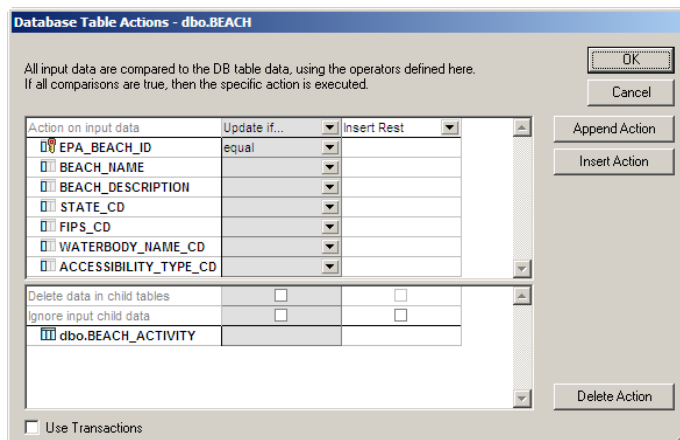
12. Click the **Output** tab at the bottom of the screen to preview your transformation. Note that MapForce generates INSERT and UPDATE statements to update/append records defined in the XML file into the target database.

13. Save the mapping file as **XMLtoDB.mfd** to **C:\MapForceTutorial\XMLtoDB**.

## 2.3.2   Generate Code and Assemble the Plugin

Please see the appropriate section below for either .NET or Java.

2.3.2.1   Microsoft .NET

6. Determine whether you will be generating code for Visual Studio 2005 or 2008. By default, MapForce will generate a Visual Studio 2008 project. To modify this setting, select the **Tools** menu, choose **Options**, select the **Generation** tab, and choose the appropriate code generation setting.

7. From the **File** menu, choose **Generate Code in -> C# (sharp)**.

8. When prompted, choose the **C:\MapForceTutorial\XMLtoDB** folder and click **OK**. The Visual Studio solution will be generated in the selected folder.

9. Open the Mapping.sln located in **C:\MapForceTutorial\XMLtoDB\Mapping** using Visual Studio 2005 or 2008. Next, compile the project by selecting **Build Solution** from the **Build** menu.

> Note that MapForce generates a sample console project named **Mapping**. While not necessary for the purposes of this tutorial, it is interesting to examine the code in **MappingConsole.cs** to see how the transformation is invoked and how it can be tested from the command line.

10. Navigate to the **C:\MapForceTutorial\XMLtoDB\Mapping\bin\Debug** folder to view the compiled output.
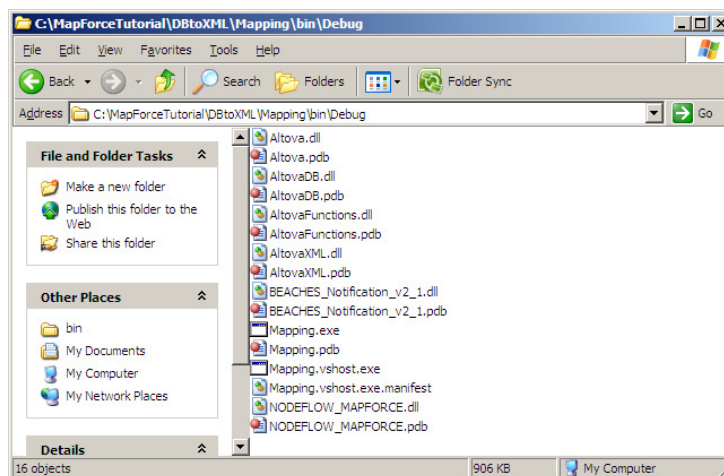
11. Visit the <u>OpenNode2 code repository</u> and download the compiled .NET MapForceBridge Plugin.

12. Extract the downloaded plugin files to a directory on your computer. Copy the contents of the folder to the clipboard.

13. Navigate back to **C:\MapForceTutorial\XMLtoDB\Mapping\bin\Debug** and paste the plugin files into the directory alongside the compiled MapForce output. The MapForce Bridge plugin files will be added to the directory as shown in the image below:
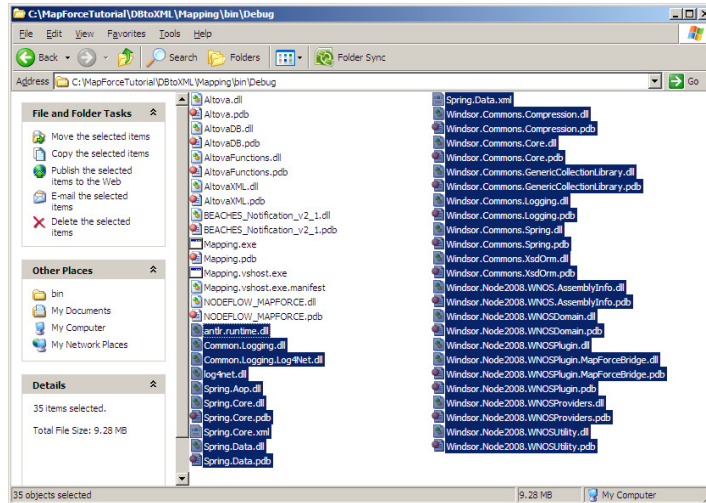
14. Press **CTRL+A** to select all the files, right-click on any file and choose **Send to -> Compressed (zipped) Folder**.

15. Rename the newly-created ZIP file **MapForceSubmitTutorial.zip**. (optional step)

You have successfully created the plugin and are now ready to install it to OpenNode2. Please skip to section 2.2.4 to continue.

2.3.2.2   Java

7. From the **File** menu, choose **Generate Code in -> Java**.

8. When prompted, choose the **C:\MapForceTutorial\XMLtoDB** folder and click **OK**. The Java source code will be generated in the selected folder.

9. Compile the Java plugin using the generated ANT build script. The final output should be a single **Mapping.jar** file.

10. If you did not already download the MapForceBridge plugin in the previous tutorial, visit the <u>OpenNode2 code repository</u> and download the compiled Java MapForceBridge Plugin (**mapforcebridge.jar**).

11. Create a single ZIP file containing both **Mapping.jar** and **mapforcebridge.jar**.

12. Name the zip file **MapForceSubmitTutorial.zip**. (optional step)

You have successfully created the plugin and are now ready to install it to OpenNode2.

2.3.3   Install and Configure the Plugin

Now that the plugin is built, it can now be installed on OpenNode2.

1. Log into the OpenNode2 Admin Utility.

2. <u>**Java Node only:**</u> If you did not already create a data source for the previous tutorial, follow the directions in section 2.2.4, step 2 to create the data source for the BEACHDEMO database/schema.

<u>Add the BEACHDEMO_IN Exchange</u>

3. Follow steps 3-7 in section 2.2.4 to create a new exchange, only name the exchange **BEACHDEMO_IN**.

> **Note**: A new exchange must be created since a MapForce-based plug-in can only support a single Query/Solicit service or Submit service.

<u>Upload the Plugin</u>

1. Next, click **Upload Plugin** from the left-side navigation menu.

2. Click the **Browse…** button and select the **MapForceSubmitTutorial.zip** file created in the previous section.

3. Choose **BEACHDEMO_IN** from the **Exchange** drop down box.

4. Click **Upload** to complete uploading the plugin.

<u>Add the Submit Processor Service</u>

Next, a service will be added that will allow the node to process an inbound XML submission to OpenNode2.

5. On the Data Exchange Manager tab, click **Add Service**, located beneath the BEACHDEMO_IN exchange header.

6. In the **Service** field, enter an asterisk (*) for the service name.

> **Note**: The asterisk name makes the service compatible with the OpenNode2's v1.1 endpoint. If a specific name is given, the service will only be compatible with the v2.0 endpoint since the v2.0 specification requires a **flowOperation** to be provided by the submitter. See the OpenNode2 documentation for more information.

7. Select the **MapForceBridge** item from the **Implementer** drop down box. (This should be the only choice available)

    a. **Java Node only**: Click the **Next** button

8. Select **Submit** from the **Type** drop down box. This tells OpenNode2 to use the MapForceBridge implementor to process Submit operations to the **BEACHDEMO_IN** exchange.

9. Perform the following step for your specific environment:

    b. **.NET Node only:** Type the connection string to the database in the **Database Connection String** field. This will be passed to the MapForce mapping code to establish a connection to the source database.

        > **Note:** For .NET users, the connection string can be found in the MapForce-generated **MappingConsole.cs** file.

    c. **Java Node only:** In the Source **Data Provider** drop down box, select the BEACHDEMO_DB connection created at the beginning of this section.

10. Click **Save** to save the new service.

2.3.4   Test the Service

Lastly, you will submit a Beach Notification v2.1 schema compatible XML file to OpenNode2. OpenNode2 will insert/update data in the database based on the contents of the XML file.

1. Launch NodeClientLite2.

2. Connect to your node from the **Connect to Node** screen by either typing in your node URL or selecting the node from the Connection History list. Click **Connect** to initiate the connection.

Submit the XML file to OpenNode2

3. Click Upload Documents from the Things I can do navigation menu. The Upload Documents screen will appear:
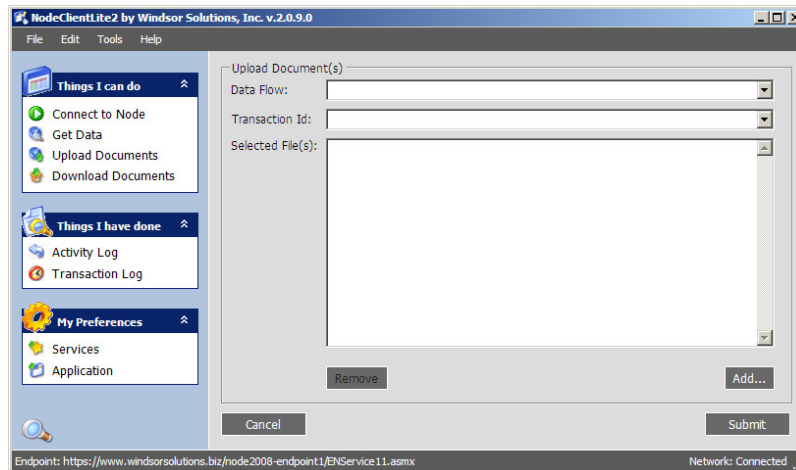
4. Type **BEACHDEMO_IN** in the Data Flow field.

5. Click the **Add…** button and browse to the **Beach_Submit_Example.xml** file located in the **C:\MapForceTutorial\XMLtoDB** folder and click **OK**.

6. Click **Submit** to submit the file to OpenNode2.

Examine OpenNode2 Activity Log (optional)

Next, you will examine the Activity Log in OpenNode2 to view the processing activity.

7. Log into OpenNode2 and click the **Activity** tab.

8. Filter the activity log for the **BEACHDEMO_IN** exchange and click **Search**.

9. An **Info (Submit)** entry should display. Expand the entry by clicking the button. The log detail will display similar to the image below:

If the file processed correctly, the transaction status should be set to "Processed".

Examine the New/Updated Database Data

Lastly, you will examine the database tables to see that new records have been added.

1. Open the BEACH table and note that a new beach named "My New Beach" has been added:

	EPA_BEACH_ID	BEACH_NAME	BEACH_DESCR...	STATE_CD	FIPS_CD	WATERBODY_...	ACCESSIBILITY...
▶	MI000286	Thompson Park	Thunder Bay	MI	32001	LAKE_HURON	PUB_PUB_ACC
	MI000309	Traverse City Pa...	Traverse City...	MI	32002	LAKE_MCHGN	PUB_PUB_ACC
	MI000999	My New Beach	This beach is def...	MI	32003	LAKE_HURON	PUB_PUB_ACC
✳	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table - dbo.BEACH_ACTIVITY   Table - dbo.BEACH   Summary

2. Open the BEACH_DETAIL table and note that two new beach advisories have been added:

	ACTIVITY_ID	EPA_BEACH_ID	ACTIVITY_TYPE	ACTIVITY_NAME	START_DATE	REOPEN_DATE	REASON_TYPE...	POLLUTION_S...	INDICATOR_TY...
▶	1111	MI000286	CLOSURE	Beach Closure	6/1/2009 12:00:...	6/3/2009 12:00:...	ELEV_BACT	RUNOFF	ECOLI
	2222	MI000286	CLOSURE	Beach Closure	6/6/2009 12:00:...	6/7/2009 12:00:...	ELEV_BACT	RUNOFF	ECOLI
	5dc0b62814c94...	MI000286	CLOSURE	Beach Closure	6/10/2009 12:0...	6/11/2009 12:0...	ELEV_BACT	RUNOFF	ECOLI
	80565a4c14154...	MI000999	CLOSURE	Beach Closure	7/1/2009 12:00:...	7/3/2009 12:00:...	ELEV_BACT	SEPTIC	ECOLI
✳	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table - dbo.BEACH_ACTIVITY   Table - dbo.BEACH   Summary

Congratulations, you have successfully completed the MapForceBridge Plugin tutorial!