# BTSHOL03: Working with Maps

Objective

After completing this lab, you will be able to:

Create a map by using BizTalk Mapper.

Add functoids to a map.

Validate a map.

Build a map project.

Debug a map

Scenario

As part of a business-to-business (B2B) process, Northwind Traders needs to send orders to a supplier named Fabrikam. Northwind Traders uses a file format for customer orders that is different from the purchase order format that Fabrikam, requires. Because of this, you must create a map to transform Northwind Trader’s customer order schema format to the purchase order schema format required by Fabrikam.

In this lab, you will create a map to associate specific fields in a customer order (the source schema) to corresponding fields in Fabrikam’s purchase orders (the destination schema).

Estimated time to complete this lab:

Exercise 1-5: 30 minutes

Optional:   
15-20 minutes

You will then add functoids to the map to perform special transforms between fields in the source schema and destination schema. You will validate the map, and then build the project to compile the map and associated schemas and to confirm that there are no errors in the project. Finally, you will use map debugging to find and fix a problem with the map based on production data.

User Name: **Administrator**

Password: **pass@word1**

Exercise 1  
Creating a Schema Map using the BizTalk Mapper

In this exercise, you will use BizTalk Mapper to create a map. This map will be used to create links that associate the data fields between two different schemas.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Open an existing solution.  This opens a project with predefined schemas (.xsd). | 1. On the **Start** menu, click **Microsoft Visual Studio 2015**. 2. On the **File** menu, point to **Open**,and then click **Project/Solution**. 3. In the **Open Project** dialog box, browse to **C:\Labs\Lab 3\Start\NWBusinessSolution**, click **NWBusinessSolution.sln**, and then click **Open**.   The existing project opens in Solution Explorer. |
| Add the destination schema to the project.  This loads the schema for the purchase order format used by Fabrikam. | 1. In Solution Explorer, right-click the **NWMessaging** project, and then click **Add Reference**. 2. In the **Add Reference** dialog box, on the **Browse** tab, browse to **C:\Labs\Lab 3\Start**, click **FKMessaging.dll**,and then click **Add**. 3. Click **OK**.   The FKMessaging reference is added to the Project. |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Create the BizTalk map.         Selecting the **Map** template causes BizTalk Mapper to start after the map is added to the project. | 1. In Solution Explorer, right-click the **NWMessaging** project, point to **Add**, and then click **New Item**. 2. In the **Add New Item** dialog box, in the Categories pane, click **Map Files**, and then in the Templates pane, click **Map**. 3. In the **Name** box, type **NWCustomerOrder\_To\_FKSupplierPO.btm** to name the map.      1. Click **Add** to start BizTalk Mapper.   BizTalk Mapper opens and the new map is added to Solution Explorer. |
| Open the source and destination schemas.  This prepares the Mapper’s UI to translate the source schema to the destination schema. | |  |  | | --- | --- | | Source schema | Destination schema | | Schemas  NWMessaging.CustomerOrder | References  FKMessaging  Schemas  FKMessaging.SupplierPO |  1. In BizTalk Mapper, click the links in the Source Schema view and the Destination Schema view to select the following schemas: |
| Rename the map page.   Renaming the map page makes map management easier. It does not affect the XSLT in any way. | 1. Below the Map Zone, right-click the **Page 1** tab (at the bottom of the map), and then rename the page **Data Maps**. |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Create a link between fields.  Linking nodes in the Mapper creates XSLT that will transform the data in the source document into the structure of the destination document. | 1. mappUsing the pairings in the table below, in the Source Schema pane, click each source field and then drag it across the Mapper grid to the associated destination field in the Destination Schema pane.   To expand all nodes in the schema right-click the <Schema> node, and then click Expand Tree Node.   |  |  | | --- | --- | | Source Field | Destination Field | | CustomerPONumber | PONumber | | UnitPrice (under SalesOrderLine) | Cost (under Items\Item) | | Quantity | Qty | | ItemID | ProdID | | Status | Status |   Your map should look similar to this: |

Exercise 2  
Adding Functoids to a Map

In this exercise, you will add functoids to transform a value from a field in the source schema to a field in the destination schema. Functoids provide processing that allows for conditional or advanced transformation of data beyond copying a value from the destination.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Add a cumulative functoid to the map.  This functoid calculates the total quantity of all line items in the customer order before updating the total quantity in the purchase order. | 1. If the Toolbox is not docked on the left side, on the **View** menu, click **Toolbox**. 2. In the Toolbox, on the **Cumulative Functoids** palette, click the **Cumulative Sum** functoid, and then drag it to the Mapper grid. 3. In the Source Schema pane, click the **Quantity** field, and then drag a link to the **Cumulative Sum** functoid in the Mapper grid. 4. In the Mapper grid, click the **Cumulative Sum** functoid, and then drag a link to **TotalQty** in the Destination Schema pane. |
| 1. Add a multiplication functoid to the map.   This functoid calculates the total cost of a line item.   The output of this functoid will not go directly to the destination document. Instead, this functoid will be used as input to another functoid. | 1. In the Toolbox, on the **Mathematical** **Functoids** palette, click the **Multiplication** functoid, and then drag it to the Mapper grid. 2. In the Source Schema pane, click the **UnitPrice** field, and then drag a link to the **Multiplication** functoid in the Mapper grid. 3. In the Source Schema pane, click the **Quantity** field, and then drag a link to the **Multiplication** functoid in the Mapper grid. |
| 1. Add another cumulative functoid to the map.   This functoid calculates the total cost of all line items in the customer order before updating the total cost in the purchase order. | 1. In the Toolbox, on the **Cumulative Functions** palette, click the **Cumulative Sum** functoid, and then drag it to the Mapper grid.   Make sure to place this functoid to the right of the multiplication functoid. Functoids must always link from left to right.   1. In the Mapper grid, drag a link from the **Multiplication** functoid to the **Cumulative Sum** functoid you just added. 2. In theMapper grid,drag a link from the **Cumulative Sum** functoid to the **TotalCost** field in theDestination Schema pane. |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Add a scripting functoid to a map.   This step changes the value in the Priority field from an integer to a string.  This function could be written in C#, Jscript .NET, XSLT, or an external assembly. | 1. In the Toolbox, on the **Advanced Functoids** palette, click the **Scripting** functoid, and then drag it to the Mapper grid. 2. In the Source Schema pane, click the **Priority** field, and then drag a link to the **Scripting** functoid in the Mapper grid. 3. Click the **Scripting** functoid, and then drag a link to the **Priority** field in the Destination Schema pane. 4. Right-click the **Scripting** functoid, and then click **Configure Functoid Script**. 5. In the **Configure Functoid Script** dialog box, in the **Script type** list, choose **Inline Visual Basic .NET**, and then copy the code from **C:\Labs\Lab 3\Start \ScriptingFunctoid.txt** to the **Inline script** **buffer** text box (overwrite any code already present):  |  | | --- | | *Public Function PriorityConversion(ByVal param1 As String) \_*  *As String*  *dim priority as string = ""*  *Select param1*  *Case “1”*  *priority = “High”*  *Case “2”*  *priority = “Medium”*  *Case “3”*  *priority = “Low”*  *End Select*  *Return priority*  *End Function* |  1. Click **OK** to save the script.   The completed map should look similar to this: |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create a new map page.  You are not required to create a new page, but it will aid in readability. | 1. Below the Map Zone, right-click the **Data Maps** tab, click **Add Page** to create a new page (or map zone), and then rename the page as **Control**. |
| 1. Add a **Looping** functoid to the map.  This functoid loops through all occurrences of line items in the customer order and creates multiple line items in the purchase order.   If no **Looping** functoid is present, the Mapper will try to determine how many of the destination nodes to create for each occurrence of the source node. Even if the Mapper chooses the looping that you desire, it is best for documentation purposes to explicitly use the **Looping** functoid. | 1. In the Toolbox, on the **Advanced** **Functoids** palette, and then drag the **Looping** functoid to the Mapper grid on the Control page. 2. In the Source Schema pane, click the **SalesOrderLine** record, and then drag it to **Looping** functoid in the Mapper grid. 3. In the Mapper grid, drag a link from the **Looping** functoid to the **Item** record in the Destination Schema pane.   The completed map should look like this:   1. On the **File** menu, click **Save All**. |

Exercise 3  
Validating and Testing the Map

In this exercise, you will use the **Validate Map** command to determine whether the map contains any internal inconsistencies, or has other issues that might prevent it from being used effectively for mapping schemas.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Validate the map.  This makes sure that the map could produce a valid destination document.  An example of an invalid map would be one in which a required field in the destination document is not filled. | 1. In Solution Explorer, right-click **NWCustomerOrder\_To\_FKSupplierPO.btm**, and then click  **Validate Map**.   The results of the map validation are displayed in the Output window. A link to the generated XSLT file is also provided to aid in advanced debugging scenarios. |
| 1. Test the map.  Testing a map provides a way to catch exceptions or logical errors by using an actual instance document without deploying an entire solution. | |  |  | | --- | --- | | Property | Value | | **TestMap Input Instance** | C:\Labs\Lab 3\Start\CustomerOrder.xml | | **TestMap Input** | XML |  1. Click **NWCustomerOrder\_To\_FKSupplierPO.btm**, and in the **Properties** window set these properties with the values shown: 2. Click **OK** to save the properties. 3. Right-click **NWCustomerOrder\_To\_FKSupplierPO.btm**, and then click **Test Map**.   A link to the XML instance is shown in the Output window.   1. In the Output window, click the link while pressing **CTRL** to open the resulting XML file. (If an Internet explorer dialog appears, click OK to dismiss it)   Note that the priority should be a text value, converted by the scripting functoid from an integer in the source, and the cumulative functoids have calculated total values in several places. |

Exercise 4  
Building the Map Project

In this exercise, you will build the project to generate an assembly that contains the resources (the schemas and the map) that you have created. This also ensures that that there are no compile-time errors in the work you have completed so far.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Build the project.  The assembly is compiled into a DLL file and saved in the …\bin\Debugfolder within the project folder. | 1. On the **File** menu, click **Save All**. 2. In Solution Explorer, right-click **NWMessaging**, and then click **Build**.   The results of the compiled project are displayed in the Output window. |

Exercise 5  
Debugging a BizTalk map using the BizTalk Mapper

The staff at Northwind have been testing their map files in Visual Studio 2015 and do not receive any errors or warnings, but in the production system they are getting reports of incorrect results in the purchase orders created by their map. In this exercise, you will test the existing map to help discover the problem that has been occurring in the production system. You will use the new debugging features of BizTalk Server 2016 integrated into Visual Studio 2015 to find and resolve the issue.

The specific problem being reported by the customer service staff is that some purchase orders are missing the priority field. Customers are complaining as their orders, which they ranked as high priority, are not being fulfilled with the appropriate level of response.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Setup the map to use a sample file from production. | 1. In Solution Explorer, click the **NWCustomerOrder\_To\_FKSupplierPO.btm** file in the NWMessaging project. 2. In the properties dialog, edit the **TestMap Input Instance** property and select the **ProductionProblem\_CustomerOrder.xml** file found in the **Start** directory for this lab.   The map debugging feature is most useful when you have a configured input instance to use for testing. |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Test the map to verify the bug is happening in this instance. | 1. In Solution Explorer, right-click the **NWCustomerOrder\_To\_FKSupplierPO.btm** file, and then click Test Map. 2. In the output window, ensure that no errors or warnings are reported. 3. While holding the CTRL key, click the link in the output window to view the output document. 4. Find the **Priority** element, near the end of the document, and verify that it does not have a value.   The priority should contain a value of “Low”, “Medium”, or “High” based on a value specified in the input document. |
| 1. Set breakpoints in the XSLT file to aid in your debugging. | 1. In Solution Explorer, right-click the **NWCustomerOrder\_To\_FKSupplierPO.btm** file then click **Validate Map**. 2. In the output window find the link preceded by the text “The output XSLT is stored in the following file:”. 3. While holding the CTRL key down, click the link to view the XSLT document. 4. Right-click inside the document window and choose **View Source** from the context menu.   This is necessary because by default the XSL file is opened using the browser window instead of the text editor.   1. Within the document window, look for the following text (you will have to scroll down a little to find it):   <xsl:variable name="var:v8"  select="userVB:PriorityConversion(  string(Priority/text()))" />  <Priority>  <xsl:value-of select="$var:v8" />  </Priority>   1. In the left margin, click next to each of the first two lines to create breakpoints on the two lines most relevant to your current defect.   XSLBreakpoints |
| 1. Use the debugger to find the problem. | 1. Save and close all documents. 2. In Solution Explorer, right-click on the **NWCustomerOrder\_To\_FKSupplierPO.btm** file and choose **Debug Map** from the context menu.   At this point, two document windows should be open: the XSL document and the output XML document being created. If you look at the output XML document it will be empty at this point. If prompted to replace the output XML file, select Yes.   1. In the XSL document, ensure the breakpoints you set in the last step are still present. 2. Press **F5** (or choose **Continue** from the **Debug** menu) and the execution should stop at your first breakpoint.   At this point another document window will open showing the input XML document as well.   1. Press the **F11** key (or choose **Debug** | **Step Into**) and the debugger should step into the Visual Basic script function that is used to set the variable var:v8. 2. Use the **F10** (**Debug | Step Over**) command to step through the select scope in the function, BUT DO NOT EXIT THE FUNCTION. 3. Before exiting the function, open the **Locals** window and find the **param1** variable to check its value and find out why it is not being matched in the select statement.   Notice that the param1 being passed to the function has a value of “01” instead of an integer value between 1 and 3 as expected by the function.   1. Continue to step out of the function and then check the **Locals** window again to check the value of the **var:v8** variable.   Notice that the function returned an empty string, which is being used as the value of the Priority element.   1. Now that you have found the defect, press **F5** (**Debug | Continue**) to complete the debugging session so you can fix the problem. |
| 1. Fix the problem and debug again to ensure the fix worked. | 1. Open the **NWCustomerOrder\_To\_FKSupplierPO.btm** file by double-clicking it in Solution Explorer. 2. Right-click the scripting functoid, connected to the Priority element in the output schema, and choose **Configure Functoid Script** from the context menu. 3. Edit the Visual Basic function to so that the Select statement returns “Error” if a value outside the expected range is received as input as shown in bold below.   *Public Function PriorityConversion(ByVal param1 As String) \_*  *As String*  *dim priority as string = ""*  *Select param1*  *Case “1”*  *priority = “High”*  *Case “2”*  *priority = “Medium”*  *Case “3”*  *priority = “Low”*  ***Case Else***  ***priority = "Error"***  *End Select*  *Return priority*  *End Function*     1. Close the script editor and save the map file. 2. Debug the map again using the steps above and step through the Visual Basic function to ensure that the Else statement is executed for this case. 3. Inspect the output document to ensure that the results are correct. |

Optional 1  
Adding Conditional Mapping

Now suppose that you for some reason wanted to include only SalesOrderLines with a specific ItemID, to do that we will conditionally map only those elements to the destination document.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Add conditional mapping | 1. In the Solution explorer, select the **NWCustomerOrder\_To\_FKSupplierPO.btm** map. 2. In the **Properties** window, restore the **TestMap Input Instance** to *C:\Labs\Lab 3\Start\CustomerOrder.xml*. 3. In either Internet Explorer, Visual Studio or another editor, open the CustomerOrder.xml file and view its contents.   Notice the included items and their associated ItemID values.   1. Open the **NWCustomerOrder\_To\_FKSupplierPO.btm** map, and select the **Control** tab. 2. From the **Toolbox**, expand the **Logical Functoids** and add an **Equal Functoid** to the mapper grid. 3. Connect the **ItemID** node in the source document to the Equal functoid. 4. Open the Equal functoid by double clicking on it. 5. For the second parameter, enter the character string “**15**” (do not include the quotes) or any other ItemID from the CustomerOrder.xml file that you wish to filter on. 6. Connect the Equal functoid to the **Item** recordin the destination schema. 7. Right-click **NWCustomerOrder\_To\_FKSupplierPO.btm**, and then click **Test Map**. 8. **Validate** that the output is as expected – that it only includes Items with ProdId 15 (or another that you selected). |

Optional 2  
Creating Custom Functoids

Read about Custom Functoids at <http://msdn.microsoft.com/en-us/library/aa560879.aspx>.

Look at the Custom Functoid sample whose description can be found at <http://msdn.microsoft.com/en-us/library/aa577632.aspx>.

Its source at C:\Program Files (x86)\Microsoft BizTalk Server 2016\SDK\Samples\XmlTools.

Focus on the implementation of the sample – the CustomFunctoid.cs class.

Put less focus on actually deploying and running the sample.

Optional 3  
Using the Power of the Scripting Functoid

Look at the Extending the Mapper sample and its use of the Scripting functoid whose description can be found at <http://msdn.microsoft.com/en-us/library/aa560433.aspx>.

Its source at C:\Program Files (x86)\Microsoft BizTalk Server 2016\SDK\Samples\XmlTools.

Focus on the information in the table towards the end of the page that begins with



and looking at the corresponding functoid in the map for how this is done.

Put less focus on actually deploying and running the sample.