BEA AquaLogic Service Bus™

Transforming Data Using the XQuery Mapper

Version: 2.1
Document Revised: December 2005
## Contents

**XQuery Mapper Overview**
- Data Transformation .................................................. 1-1
- BEA XQuery Mapper ................................................... 1-1
  - What Happened to DTF Files? ................................. 1-2

**Transforming Data Using XQuery**
- Launching the XQuery Mapper ...................................... 2-2
- XQuery Mapper Sample Project ..................................... 2-3
- Creating a New XQuery Mapper Project ......................... 2-4
- Creating and Importing Schema Files ......................... 2-4
  - Importing XML Schemas ....................................... 2-5
  - Creating XML Schema Files Using XML Schema Editor .... 2-7
  - Creating WSDL Files Using WSDL Editor .................. 2-7
  - Importing MFL Schemas ....................................... 2-8
- Selecting Source and Target Data Types to Generate XQuery 2-9
- Using the Design View to Create Data Transformations .... 2-12
  - Creating Basic Element Transformations .................... 2-12
  - Creating Basic Attribute Transformations .................. 2-13
  - Creating Complex Map Transformations ..................... 2-14
- Editing Map Transformations ..................................... 2-15
  - Viewing and Editing Generated XQuery Files ............. 2-16
- Using the Constraints Tab ......................................... 2-16
Using the Target Expression Tab .................................................. 2-17
Using the Target Expression Tab to Edit XQuery Code of a Link .......... 2-17
Using the Target Expression Tab to Add If-Then-Else Constructs to a Link . 2-18
Inserting Calls to XQuery Functions ........................................... 2-18
Invoking XQuery Functions in a Query ........................................ 2-19
Using Expression Variables ..................................................... 2-20
Using the Property Editor When Editing an XQuery file .................... 2-20
Testing Map Transformations .................................................... 2-21
XQuery Mapper Testing Functionality Overview ............................ 2-21
Validating During Design Time ............................................... 2-22
Understanding Design View Graphical Representations .................... 2-23
Link Menu Options ............................................................... 2-24
Understanding Map Representations ........................................ 2-25
Understanding Map Transformation Links .................................. 2-26
XML Global Elements, Global Types, Local Elements, and Attributes ... 2-29

Examples: Manipulating and Constraining Data Using XQuery
Combining Data From Different Schemas .................................... 3-1
Mapping Repeating Elements and Creating a Join .......................... 3-4
   Step 1. Create a New XQuery File ........................................... 3-4
   Step 2. Add a Conditional Constraint ...................................... 3-6
   Step 3. Add Links to Populate Empty Element ........................... 3-8
   Step 4. Calculate Total Cost ............................................... 3-9
   Step 5. Add a Constraint With Multiple Conditions ...................... 3-10
Using the Union Option of the Constraints Tab .............................. 3-12
Creating a Transformation Between a Repeating Source and Non-Repeating Target . 3-14
Creating a Transformation Between a Non-Repeating Source and Repeating Target . 3-17
Creating a Nested If-Then-Else Expression .................................. 3-22
Step 1. Create New XQuery Transformation ........................................... 3-23
Step 2. Create First If Condition. ......................................................... 3-24
Step 3. Create First Nested If-Then-Else Condition. .............................. 3-25
Step 4. Create Second Nested If-Then-Else Condition .............................. 3-26
Using Recursive Schemas ................................................................. 3-28
Creating Group By Key Fields XQuery Constructs ................................. 3-30
Testing XQuery Files ................................................................. 3-34
XQuery Mapper Overview

This section provides a brief introduction to the BEA XQuery Mapper. It provides an overview of data transformation and XQuery mapper concepts.

This section includes the following topics:

- Data Transformation
- BEA XQuery Mapper
- What Happened to DTF Files?

Data Transformation

The XQuery Mapper provides a graphical environment for data transformation. Data is transformed from one format to another and the resulting output is a query. The query is written in the XQuery language, which is defined by the World Wide Web Consortium (W3C). For more information about the W3C and the XQuery language, see http://www.w3.org/Consortium/.

BEA XQuery Mapper

The XQuery Mapper is a plug-in within the Eclipse development environment. Therefore, to use the XQuery Mapper, Eclipse must be installed and launched. Eclipse and the XQuery Mapper are provided as part of the product installation. For information about starting the XQuery Mapper, see “Launching the XQuery Mapper” on page 2-2.

Using the XQuery Mapper, XML and non-XML data can be transformed from one format to another. For example, XML that is valid against one schema can be converted to XML that is
valid against a different schema. The source and target data of a transformation can be XML or non-XML. The data format files can be .xsd (XML Schema), .wsdl (Web Service Definition Language), or .mfl (Message Format Language) files. All these files can be “Creating and Importing Schema Files” on page 2-4. MFL documents are created using the Format Builder tool. For more information, see the Format Builder Online Help.

A data transformation can have multiple input types but only one target type. For example, data can be transformed from two sources to one target, as shown in the following figure.

**Figure 1-1  Data Transformation Overview**

What Happened to DTF Files?

In the BEA WebLogic Integration 8.1 release, data transformations were created in BEA WebLogic Workshop IDE and stored in Data Transformation Files (also known as DTF files). These transformation files contained a single data transformation, which represented the mapping and conversion of data from one format to another.

In this release, data transformations are created using the XQuery Mapper and the resulting query is an XQuery file (saved with an .xq extension).
Topics Included in This Guide

Chapter 2, “Transforming Data Using XQuery”
Describes how to use the XQuery Mapper to create a query that is written in the XQuery language.

Chapter 3, “Examples: Manipulating and Constraining Data Using XQuery”
Provides information on how to use the XQuery Mapper Sample Project. Procedures for using the sample schemas are included.
This section describes how to use the BEA XQuery Mapper to graphically create data transformations. Using the graphical interface, you can map source elements in schemas to target elements in schemas. After doing so, the XQuery Mapper generates an XQuery, which is saved as an .xq file.

This section includes the following topics:

- Launching the XQuery Mapper
- XQuery Mapper Sample Project
- Creating a New XQuery Mapper Project
- Creating and Importing Schema Files
- Selecting Source and Target Data Types to Generate XQuery
- Using the Design View to Create Data Transformations
- Editing Map Transformations
- Using the Constraints Tab
- Using the Target Expression Tab
- Inserting Calls to XQuery Functions
- Using Expression Variables
- Using the Property Editor When Editing an XQuery file
Transforming Data Using XQuery

- Testing Map Transformations
- Validating During Design Time
- Understanding Design View Graphical Representations
- XML Global Elements, Global Types, Local Elements, and Attributes

Launching the XQuery Mapper

When you install BEA AquaLogic Service Bus, Eclipse 3.1 and the XQuery Mapper plug-in for Eclipse are installed.

Note: For the AquaLogic Service Bus 2.1 release, Eclipse 3.1 is supported only on Windows platforms.

As part of your XQuery Mapper installation, a sample project containing default schema is provided. For more information about the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

To Launch Eclipse

1. Start the XQuery Mapper in one of the following ways:
   - From the Windows Start Menu, choose
     Programs -> BEA Products -> Tools -> XQuery Mapper
   - By invoking the start script. To launch the XQuery Mapper in this way, BEA recommends that you double-click the following command:
     BEA_HOME/weblogic90/common/bin/eclipse.cmd
     and not BEA_HOME/eclipse31/eclipse.exe.
     Using eclipse.cmd ensures that the appropriate JDK is used.

2. From the Eclipse menu bar, open the XQuery Transformation Perspective by choosing
   Window -> Open Perspective -> Other -> XQuery Transformation.

Note: The XQuery Transformation Perspective launches automatically when you open an XQuery file. However, if the XQuery Mapper is open (and an XQuery file is not) you must manually launch the XQuery Transformation Perspective.
XQuery Mapper Sample Project

After launching the sample project, you can use the sample schema files to create XQuery Transformations as described in Chapter 3, “Examples: Manipulating and Constraining Data Using XQuery.”

To learn more, see the following topics:

- Creating a New XQuery Mapper Project
- Creating and Importing Schema Files
- Examples: Manipulating and Constraining Data Using XQuery

To Launch the Sample Project

You can open the Sample Project from within Eclipse.

- From the Eclipse menu bar, choose Help→Welcome. Then select Samples followed by Data Transformation Samples.

The Sample Project contains the following folders and files:

- **Schemas**: Contains the sample project XSD files.
- **XML**: Contains test XML files required by some of the examples described in Chapter 3, “Examples: Manipulating and Constraining Data Using XQuery.”
- **XQuery Transformations**: The folder in which you will create the XQuery files for the examples described in Chapter 3, “Examples: Manipulating and Constraining Data Using XQuery.”

Before creating XQuery transformations, you can create or import your project specific XML schema, WSDL, or MFL files. You can import files from any location to the XQuery Mapper. Before importing files into Eclipse, it is recommended that you create a folder directory structure that meets your business needs. For more information, see “Creating and Importing Schema Files” on page 2-4.
Creating a New XQuery Mapper Project

You create a new project using the **Eclipse New Project** wizard. This wizard prompts you to select from a variety of new project types. For XQuery Mapper purposes, you can choose **Simple**, which allows you to create a basic project.

**To Create a New XQuery Mapper Project**

1. Launch the XQuery Mapper Perspective in Eclipse as described in “Launching the XQuery Mapper” on page 2-2.

2. Choose **File→New→Project**.
   
   The New Project wizard opens. The wizard prompts you to select one of the following:

   - **CVS**: Using this option you can import a project from an existing source control system.
   - **Simple**: Using this option you can import a project from an existing file system.

3. Choose **Simple→Project**.

4. Click **Next**.

5. Enter a Project Name.
   
   For example, **TestProject1**

6. Verify that the **Use Default** check box is selected.

7. Click **Finish**.

**Creating and Importing Schema Files**

Schema files can be created or imported from any location. The following schema types are supported:

- **XSD (XML Schema Definition)**: Describes and constrains the contents of XML data. Multiple namespaces are supported in the XQuery Mapper. For example, you can transform two source XML files valid against a specific namespace to another XML file valid against a third namespace. For more information about importing XML schema, see “Importing XML Schemas” on page 2-5.

- **WSDL (Web Service Definition Language)**: XML schema defined in the WSDL file can be used in the transformation.
**MFL (Message Format Language):** Describes and constrains the content of non-XML data. For example, data coming from COBOL copybooks or C structure definitions. The namespace of the MFL elements are derived from the file name of the MFL document. MFL files are created using the Format Builder tool and use the .mfl extension. For more information, see the *Format Builder Online Help.*

This section includes the following topics:

- Importing XML Schemas
- Creating XML Schema Files Using XML Schema Editor
- Creating WSDL Files Using WSDL Editor
- Importing MFL Schemas

**Importing XML Schemas**

XML schemas are created and stored outside of Eclipse. However, they can also be created using the XML Schema Editor. After schema files are imported or created, they are available in the **New XQuery Transformation** wizard as shown in the following figure. For information about creating XML Schema, see: *Creating XML Schema Files Using XML Schema Editor.*
To Import XML Schemas

1. Launch the XQuery Mapper Perspective in Eclipse, as described in “Launching the XQuery Mapper” on page 2-2.

2. In the Navigator window, select the project into which you want to import the schema.

3. Choose File→Import.

   The Eclipse Import wizard is displayed. Using this wizard, schemas can be imported from a variety of sources. The import source you select depends on where the schema is stored.

4. After choosing an Import Type, click Next.

5. Depending on the Import Type selected, browse to locate the appropriate file.

   For example, if importing from a local file system, you are prompted to browse for the locally stored file.

6. After locating and selecting the input schema files, click Finish.
Creating XML Schema Files Using XML Schema Editor

You can create an XML schema file using the XML Schema Editor that is shipped with Eclipse.

To Create an XML Schema File

1. Launch the XQuery Mapper Perspective in Eclipse, as described in “Launching the XQuery Mapper” on page 2-2.

2. In the Navigator window, select the project into which you want to create the schema file.

3. Select New→Other ...

   The New wizard appears.

4. Choose either AquaLogic Service Bus or XML wizard type.

5. Choose XML Schema, then click Next.

6. Choose the parent folder in which you want to create the schema file.

7. Enter a name for the schema file.

   For example, NewXMLSchema

8. Click Finish.

The schema file is created in the specified project. You can now specify the schema details and save the file.

Note: For more information about using the XML Schema Editor, see the XML Schema Editor Tutorial, which is available at the following URL:

http://www.eclipse.org/webtools/community/tutorials/XMLSchemaEditor/XMLSchemaEditorTutorial.html

Creating WSDL Files Using WSDL Editor

You can create a WSDL file using the WSDL Editor that is shipped with Eclipse. The XML schema defined in the WSDL files can be used in the transformation.

To Create a WSDL File

1. Launch the XQuery Mapper Perspective in Eclipse, as described in “Launching the XQuery Mapper” on page 2-2.

2. In the Navigator window, select the project into which you want to create the WSDL file.
3. Select New→Other ...
   The New wizard appears.
4. Choose either AquaLogic Service Bus or XML wizard type.
5. Choose WSDL, then click Next.
6. Choose the parent folder in which you want to create the WSDL file.
7. Enter a name for the WSDL file.
   For example, NewWSDLFile
8. Click Next.
9. Enter the target namespace and prefix of the WSDL file.
10. If required, select the Create WSDL Skeleton check box.
11. After choosing the protocol and the binding options, click Finish.
    The WSDL file is created in the specified project.

   Note: For more information about using the WSDL Editor, see the WSDL Editor Tutorial, which is available at the following URL:
   http://www.eclipse.org/webtools/community/tutorials/WSDLEditor/WSDLEditorTutorial.html

Importing MFL Schemas

After MFL files are imported, the files are available in the New XQuery Transformation Wizard.

To Import MFL Schemas

1. Launch the XQuery Mapper Perspective in Eclipse, as described in “Launching the XQuery Mapper” on page 2-2.
2. In the Navigator window, select a project into which you want to import the schema.
3. Choose File→Import.
   The Import wizard opens. Using this wizard, schema can be imported from a variety of sources. The import source you select depends on where the schema is stored.
4. Choose an Import Type, then click Next.
5. Depending on the Import Type selected, the wizard prompts you to browse for the import file.

For example, if importing from a local file system you will be prompted to browse for the locally stored file.

6. After locating and selecting the input schema files, click Finish.

To Create MFL Files

MFL files are created using the Format Builder tool, which can be launched from the Windows Start menu or from Eclipse.

- To launch Format Builder from Eclipse, select Tools → BEA → Format Builder from the Eclipse menu bar.
- To launch Format Builder from the Start menu, select Start → Programs → BEA Products → Tools → Format Builder

For more information about Format Builder, see the Format Builder Online Help.

Selecting Source and Target Data Types to Generate XQuery

Click the Design tab to open the XQuery Mapper Design view, which allows you to create transformation maps by selecting source and target data types. In the Design view, you can graphically link Source and Target elements. After creating links between source and target elements, an XQuery file (or map) is generated. The generated XQuery file has an .xq extension. The XQuery file (or map) is displayed in the Design view. The generated XQuery code is visible by clicking the Source tab.

Source and target data types can be of the following combinations:

- Non-XML to XML
- XML to Non-XML
- XML to XML
- Non-XML to Non-XML

To Select Source and Target Data Types

1. Select the project for which you are creating a map.

2. Right-click and choose New → XQuery Transformation.
The New XQuery Transformation wizard opens.

3. Enter a file name for the transformation.
   For example, Transform1

4. Click Next.
   The Source Types page opens. This page allows you to select source types for the transformation.

5. In the Available Source Types pane choose the source data type:
   – If source input data is XML, choose XML.
   – If source input data is MFL, choose Non-XML.

   Note: For schema representations to be available in the Available Source Types and Available Target Types pane, the XML and Non-XML files that contain these schemas must be imported or created.

6. Select source data elements.
   For example, if adding input data from Schemas/Dates.xsd, first select it, then traverse its contents to select date as your input element, as shown in the following figure.
Selecting Source and Target Data Types to Generate XQuery

7. After selecting the source input parameter, click **Add**.

The elements and attributes that make up the selected element are displayed in the **Selected Source Types** pane.

8. Repeat for adding more source types. Multiple source types can be specified.

9. Click **Next**.

The **Target Types** page opens. This page allows you to select the target data types for the transformation.

10. In the **Available Target Types** pane choose the target data type:

    - If target input data is XML, choose **XML**.
    - If target input data is non-XML, choose **Non-XML**.

    **Note**: For schema representations to be available in the **Available Source Types** and **Available Target Types** pane, XSD and MFL files that contain these schemas must be imported into your Eclipse project.

11. Select the target type.
Using the Design View to Create Data Transformations

This section includes the following topics:

- Creating Basic Element Transformations
- Creating Basic Attribute Transformations
- Creating Complex Map Transformations

Creating Basic Element Transformations

A basic element transformation involves mapping a source element to a target element. The source and target elements may or may not have the same name, type, or scope. There are many different types of basic element transformation.

The following list provides examples:

- **Element to Element**: An source element is mapped to a target element.
- **Element Combination**: Multiple source elements are combined to create a single target element.
- **Element Explosion**: XQuery string functions are exploded from a single source element to multiple target elements.

To Create an Element to Element Transformation

1. Select the project for which the map is being created.
2. Open the XQuery file for which you are creating a map.
   This assumes that the map file has been created as described in “Selecting Source and Target Data Types to Generate XQuery” on page 2-9.
3. To create an **Element to Element** link, drag an element from the **Source** pane to an element in the **Target** pane.
For example, if creating an **Element to Element** link between the source element `customer-name` and the target element `customer-name` simply drag it from the **Source** pane to the **Target** pane as shown in the following figure.

![Figure 2-3 Element to Element Links](image)

- While dragging from the **Source** pane to the **Target** pane, a temporary dashed line appears between the two nodes. For more information, see “Link Menu Options” on page 2-24
- After the target and source elements are connected a line (either dashed or solid) is displayed. For more information, see “Understanding Map Transformation Links” on page 2-26

4. Repeat the preceding step until all the desired elements are mapped.

5. Save your changes.

### Creating Basic Attribute Transformations

A basic attribute transformation involves mapping a source attribute to a target attribute. The source and target attributes may or may not have the same name, type, or scope. There are many different types of basic attribute transformation.

The following list provides examples:

- **Element to Attribute**: Source elements mapped to target attribute.
- **Attribute to Element**: Source attributes mapped to target element.
- **Attribute to Attribute**: Source attributes mapped to target attributes where the attribute names (in source and target) are the same.

**To Create an Attribute to Element Transformation**

1. Select the project for which the map is being created.
2. Open the XQuery file for which you are creating a map.

3. Create an **Attribute to Element** link by dragging an attribute from the **Source** pane to an element in the **Target** pane.

   For example, if creating an **Attribute to Element** link between the `street` attribute and the `street` element, drag it from the **Source** pane to the **Target** pane as shown in the following figure.

   ![Attribute to Element Link](image)

   - While dragging from the **Source** pane to the **Target** pane, a temporary dashed line appears between the two nodes. For more information, see “Link Menu Options” on page 2-24.
   - After the target and source have been connected, a line (either dashed or solid) is displayed. For more information, see “Understanding Map Transformation Links” on page 2-26.

4. Repeat the preceding step until all the desired attributes are mapped.

5. Save your changes.

### Creating Complex Map Transformations

Complex map transformations involve mapping from a complex source (for example, a repeating element) to a complex target (for example, a non-repeating element). The following list provides examples:

- **Repeating Group to Repeating Group**: The source consists of a variable number of instances of a group of elements and each source group needs to be mapped to an instance of the target group.

- **Repeating Group to Non-Repeating Element**: The source consists of a variable number of instances of a group of elements and each source group needs to be mapped to an instance of the target element.
To Create a Repeating Group to Repeating Group Transformation

1. Select the project for which the map is being created.

2. Open the XQuery file for which you are creating a map.
   This assumes that the map file has been created as described in “Selecting Source and Target Data Types to Generate XQuery” on page 2-9.

3. To create a Repeating Group to Repeating Group link, drag an element or attribute from the Source pane to an element or attribute in the Target pane.
   For example, if creating a Repeating Group to Repeating Group link between product and product, drag it from the Source pane to the Target pane as shown in the following figure.

   **Figure 2-5  Repeating Group to Repeating Group Link**

   - While dragging from the Source pane to the Target pane, a temporary dashed line appears between the two nodes. For more information, see “Link Menu Options” on page 2-24.
   - After the target and source have been connected, a line (either dashed or solid) is displayed. For more information, see “Understanding Map Transformation Links” on page 2-26.

4. Repeat the preceding step until all the desired elements and attributes are mapped.

5. Save your changes.

Editing Map Transformations

After creating the initial transformation you may want to change, update, or delete it in the Source tab.

This section contains the following topics:

- Viewing and Editing Generated XQuery Files
Using the Constraints Tab

Viewing and Editing Generated XQuery Files

After creating links between source data and target data, an XQuery file is generated to represent the relationship. To learn about the XQuery language supported with XQuery Mapper, see the following URL:

http://www.w3.org/XML/Query

To Open, View, and Edit an XQuery File

1. Select the project that contains the generated XQuery file.
2. Double-click the XQuery file that you want to edit.
3. Click the Source tab.
   The XQuery code is displayed. Invalid XQuery code is underlined in red.
4. Optionally, fix errors indicated by red underline.
   If desired, you can delete all the XQuery code in the Source view of the XQ file by removing all the XQuery source code inside the function, except the root element, and recreating your links in the Design view.
5. Save your changes.

Using the Constraints Tab

The Constraints tab of the XQuery Mapper allows you to constrain or manipulate the relationship between source and target repeating elements.

The following Constraint Type options are available in the Constraints tab:

- **Repeatability/Join**
- **Union**

**Note:** For more information about these options, see “Using the Union Option of the Constraints Tab” on page 3-12.

When creating structural links between repeating elements in the Design view, XQuery for loops are generated to iterate through the repeating elements. You can use the Where Clause Expression pane of the Constraints tab to limit or constrain the target repeating elements by adding where clauses to the XQuery for loops. In the Where Clause Expression pane of the
Using the Target Expression Tab

**Constraints** tab, you can build complex conditions for the `where` clause of the XQuery `for` loop. A complex condition is made up of conditions that are joined together by OR or AND operators, for example:

\(((\text{data}($\text{PurchaseOrderDoc/partId}) > 200 \ \text{and} \ \text{data}($\text{PurchaseOrderDoc/partId}) <= 400))\)

During run time, the `for` loop will iterate only over those repeating elements that meet the complex condition.

For a detailed example on using the **Constraints Tab**, see “Creating a Transformation Between a Repeating Source and Non-Repeating Target” on page 3-14.

**Using the Target Expression Tab**

You can view and modify the link between a source and target element using the **Target Expression** tab. Using the **Target Expression** tab, you can build complex expressions between a source and target element.

This section contains the following topics:

- Using the Target Expression Tab to Edit XQuery Code of a Link
- Using the Target Expression Tab to Add If-Then-Else Constructs to a Link
- Inserting Calls to XQuery Functions

**Using the Target Expression Tab to Edit XQuery Code of a Link**

To Edit the XQuery Code of a Link

1. Double-click the XQuery file that you want to edit.
2. In the **Design View**, select or create a link between a source and target node.
   - Keep the link selected.
3. Select the **Target Expression** tab.
   - By default, the **General** option is selected and the XQuery code for the link is displayed in the **General Expression** pane.
4. Select the **Target Expression** tab.
5. Edit the generated XQuery code.
The query is written in the XQuery language. To learn more about the XQuery code, see the following URL:

http://www.w3.org/XML/Query

Using the Target Expression Tab to Add If-Then-Else Constructs to a Link

You can add an if-then-else construct to a link using the Target Expression tab. When a query is invoked with an if-then-else, the conditions that make up the if expression are evaluated, depending on the result, different values are returned for a target node. For example, if the value of quantity source node is greater than 500 then 4554 is returned as the value of the ID target node, but if the value of quantity source node is less than or equal to 500, then 5894 is returned as the value of the ID target node, as shown in the example in the following figure.

Figure 2-6 If-Then-Else

In addition to the following procedure for adding a simple if-then-else expression to a link, a more complex example is available. For an example, see “Creating a Transformation Between a Repeating Source and Non-Repeating Target” on page 3-14.

Inserting Calls to XQuery Functions

This section describes how to insert calls to functions into a query using the Target Expression tab.

For more information about the XQuery 1.0 and XPath 2.0 functions and operators, see the following URL: http://www.w3.org/TR/2005/WD-xpath-functions-20050404/

This section contains the following topics:

- Invoking XQuery Functions in a Query
- Using Expression Variables
Invoking XQuery Functions in a Query

A set of standard W3C XQuery functions and operators are provided in the XQuery Mapper. When you create a transformation, a query is written in the XQuery language and this query does the data conversion. In the generated query, you can add standard XQuery or user-defined functions. For example, as part of your transformation you might want to convert the XML String to uppercase characters.

To View the XQuery Functions and Variables in Eclipse

1. Launch the XQuery Mapper Perspective in Eclipse, as described in “Launching the XQuery Mapper” on page 2-2.

2. To view Expression Functions, choose Window→Show View→Expression Functions.

3. To view Expression Variables, choose Window→Show View→Expression Variables.

To Invoke an XQuery Function

1. Double-click an XQuery file.

2. In the Design tab, select or create a link to add the function call.

   The link between these two nodes becomes green.

3. In the bottom pane of the Design tab, choose the Target Expression tab.

   If the Target Expression tab is not visible, from the menu bar, choose Window→Show View→Target Expression.

   In the General Expression pane, the XQuery code linking the selected target and source node is displayed and is selected. Keep this selected for the next step.

4. Press Delete to remove the XQuery code in the General Expression pane.

5. Collapse and expand the folders in the Expression Functions pane to find the desired function.

   For this example, from the String Functions folder, select the upper-case function.

6. In the Expression Functions pane, select the desired function, and drag it into the General Expression pane.

   For the following step, leave the parameter of the selected function (the $string-var parameter of the upper-case function in this example) in the General Expression pane selected.
Note: XQuery functions like `trim-left` that have been defined by BEA Systems are prefixed with `fn-bea:`. XQuery functions that are not listed in the Expression Functions pane, but are defined in the XQuery specification can be used with the `fn:` prefix.

7. Select a source parameter using one of the following options:
   - From the Source pane of the Design tab select a source element and drag-and-drop it over the parameter in the General Expression pane.
   - From the Expression Variables view pane select a source variable and drag-and-drop it over the parameter in the General Expression pane.

8. Repeat until all the parameters in the function are assigned.

9. Click Apply.

During run time for this example, the `upper-case` function will convert all the characters of the source element to upper case.

Using Expression Variables

Using the Expression Variables view of the XQuery Mapper you can access variables and their sub elements. You can drag-and-drop variables or their sub elements from the Expression Variables view into the Constraints and Target Expression tabs.

The following types of variables are displayed in the Expression Variables view:

- **Source:** The variables listed in Source section of the Expression Variables view are the source types selected for the transformation in the Configure XQuery Transformation Method pane.

- **Structural Link:** When a structural link is selected in the Design tab, the Structural Link section of the Expression Variables view lists the loop iteration variable associated with the XQuery for loop generated by the structural link. This variable is in scope for all subelements of the node with the structural link.

Using the Property Editor When Editing an XQuery file

While editing an XQuery file in the Design tab, the Property Editor allows you to view schema properties about the nodes in the current map without opening the source and target XSD or MFL files. So, you can select the element or attribute in the Source, Target, or Expression Variable pane and view the associated schema properties in the Property Editor. The Property Editor displays a read-only view of the schema properties for the selected element.
To turn on the Property Editor view, choose Window→Show View→Properties.

- To change XML Schema properties, edit the XSD file that contains the XML Schema.
- To change the schema properties for non-XML data, edit the corresponding MFL (Non-XML) file.
- Selecting a link will select the target and source elements of the link.
- To deselect a link and the target and source nodes of the link, click an empty section of the middle pane (a section with no links) between the Source and Target panes.

Testing Map Transformations

You can use the Test tab in the XQuery Mapper to verify your XQuery transformations. This section contains the following topics:

- XQuery Mapper Testing Functionality Overview
- Validating During Design Time

For more information about testing XQuery files, see “Testing XQuery Files” on page 3-34.

XQuery Mapper Testing Functionality Overview

You can use the Test view in the Mapper to validate the transformation defined by you. Select the appropriate icon in the Test view as displayed in the following figure.

Figure 2-7 Test View

The following functionality is available in the Test view:

- Test XQuery: Runs the query to convert data displayed in the Source Data pane against the mappings in the query displayed in the Results Data pane.
Transforming Data Using XQuery

- **Source Variable:** The variables listed in the **Source Variables** drop-down list are the source (input) types that are currently selected. To edit another source type change the displayed source type by selecting another source type from the drop-down list.

- **Import:** Opens a file import dialog box for selecting XML files. After importing, the validation errors are marked with a yellow warning (underline) in the text editor. Hovering the mouse over the text displays a tool tip with a warning message. Click the **Test** icon to run the transformation using the imported source data.

When you import source data it is validated against the associated schema and any warnings and errors are marked. You can import XML data for Global Types and Local Elements, however, Global Types and Local elements are not validated, therefore, no errors or warnings will be reported for invalid data. For more information about Global Types and Local Elements, see “XML Global Elements, Global Types, Local Elements, and Attributes” on page 2-29.

- **Export:** Allows you to export the edited source data to an XML file. After doing so, in the Result Data pane, the Export functionality allows you to save the results of the transformation to an XML file.

- **Generate Data:** Regenerates the source data used as input to the transformation. When the **Test** tab is launched, the XQuery Mapper generates an initial set of source sample data and displays it in the **Source Data** pane. If changes are made to the data and you want to regenerate the sample data, click **Generate Data**. For example, you might want to start fresh with sample data if edits result in an XML document that is no longer valid against the associated XML Schema.

- **Auto Validate:** Select to have test files validated against source or target schema. In the source view, the Auto Validate option can be permanently turned on, or you can select it on a case-by-case basis.

**Validating During Design Time**

During design time, the **Auto Validate** icon in the **Source Data** and the **Validate** icon in the **Result Data** panes in the **Test View** tab of an XQuery file will be enabled only if the selected source parameter or resulting data is a typed global XML element. To learn more, see “XML Global Elements, Global Types, Local Elements, and Attributes” on page 2-29.

Validate will not be enabled if the selected source parameter or resulting data is one of the following types:

- **Typed Non-XML:** Untyped Non-XML (RawData) data cannot be used in transformations.
Understanding Design View Graphical Representations

- **XML global type:** For more information, see “XML Global Elements, Global Types, Local Elements, and Attributes” on page 2-29.

- **XML local element:** For more information, see “XML Global Elements, Global Types, Local Elements, and Attributes” on page 2-29.

If you select the *Auto Validate* icon in the Source Data and the Validate icon in the Result Data panes in the Test View tab of an XQuery file, the displayed XML is checked against the schema and any errors are reported during design time. The validation done during design time in the Test View is not the same as the schema validation that occurs during run time. The validation during design time does *not* modify the resulting XML document, but it does check if any required elements or attributes defined in the schema are not present.

**Understanding Design View Graphical Representations**

Using the design view you can graphically create, alter, or update transformations. This sections provides information about the graphical representations found in the Design tab of the XQuery Mapper.

This section includes the following topics:

- **Link Menu Options**
- **Understanding Map Representations**
- **Understanding Map Transformation Links**
## Link Menu Options

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Invoke by ...</th>
<th>Result ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete All Links</td>
<td>In the <strong>Design View</strong>, right-click on an empty section between the <strong>Source</strong> and <strong>Target</strong> panes and from the drop-down menu select <strong>Delete All Links</strong>.</td>
<td>In the <strong>Design View</strong>, the graphical representations of all the links are deleted and in the <strong>Source View</strong> the generated XQuery code linking the source nodes to the target nodes is deleted.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Right-clicking on an empty section between the <strong>Source</strong> and <strong>Target</strong> panes causes all the nodes to be un-selected.</td>
<td></td>
</tr>
<tr>
<td>Induce Map</td>
<td>In the <strong>Design View</strong>, right-click on a structural link and from the drop-down menu select <strong>Induce Map</strong>.</td>
<td>Data links or data structural links will be created between the child nodes of the selected structural link if source and target child elements of the link are the same subschema type. The target and source child elements must have the same name, must be the data types, and must be in the same order for the <strong>Induce Map</strong> option to create child links.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This menu option is only available when a structural link is selected in the <strong>Design View</strong>.</td>
<td></td>
</tr>
<tr>
<td>View Code</td>
<td>In the <strong>Design View</strong>, right-click on a link and from the drop-down menu select <strong>View Code</strong>.</td>
<td>The view changes from the <strong>Design View</strong> to the <strong>Source View</strong> and generated XQuery code between the links is displayed. The XQuery code for the selected link is highlighted in blue.</td>
</tr>
<tr>
<td>Disable Target Node</td>
<td>In the <strong>Design View</strong>, right-click on a link and from the drop-down menu select <strong>Disable Target Node</strong>.</td>
<td>Blocking XQuery code is added around the XQuery code of the selected link that prevents the XQuery code for the link from being executed during run time.</td>
</tr>
</tbody>
</table>
Understanding Map Representations

A data link directly transforms data from a source node to a target node. For example, the following figure shows a data link between the priceQuote/customerName element and the quote/name element.

**Figure 2-8 Data Links**

Both priceQuote/customerName and quote/name are XML String elements. During run-time, the data from the priceQuote/customerName element is converted to the quote/name element as shown in the preceding figure.

If you modify the XQuery code linking these two elements, the link between these elements changes from a data link (represented as a blue line) to an implied link (represented as a light gray line).

The following table summarizes the different link representations.
Transforming Data Using XQuery

Understanding Map Transformation Links

While dragging a node from the Source pane to the Target pane, a temporary link (a dashed line) appears between the two nodes. The color of the dotted line changes depending on the compatibility between the source and target node, as shown in the following table.

<table>
<thead>
<tr>
<th>Link Type</th>
<th>Is the Link an XQuery Mapper Generated Link?</th>
<th>Description</th>
<th>Is the Link Currently Selected in the XQuery Mapper?</th>
<th>Representation of Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Link</td>
<td>Link is generated by dragging-and-dropping.</td>
<td>A link that converts the value of the source node directly to the value of the target node.</td>
<td>Not Selected</td>
<td><img src="image" alt="Data Link" /></td>
</tr>
<tr>
<td>Structural Link</td>
<td>Link is generated by dragging-and-dropping.</td>
<td>A link between two parent structures that does not map data directly.</td>
<td>Not Selected</td>
<td><img src="image" alt="Structural Link" /></td>
</tr>
<tr>
<td>Data Structural Link</td>
<td>Link is generated by dragging-and-dropping.</td>
<td>A data structural link is the combination of the following two links:</td>
<td>Not Selected</td>
<td><img src="image" alt="Data Structural Link" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A data link between two nodes—a link that converts the value of the source node directly to the value of the target node.</td>
<td>Selected</td>
<td><img src="image" alt="Selected Data Structural Link" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A structural link—a link between two structures.</td>
<td>Selected</td>
<td><img src="image" alt="Selected" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> The link between the optional child nodes of a repeating element.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2-26
Transforming Data Using the XQuery Mapper
<table>
<thead>
<tr>
<th>Link Type</th>
<th>Is the Link a Mapper Generated Link?</th>
<th>Description</th>
<th>Is the Link Currently Selected in the Mapper?</th>
<th>Representation of Link</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constraint Link</strong></td>
<td>Link is generated by dragging-and-dropping.</td>
<td>A link that constrains or limits the resulting data of a join between source parent structures. The constraint link is created with two source nodes. <strong>Example:</strong> You could add a constraint link to a join of two source repeating elements to only return the data when the values of a particular source element are equal to each other.</td>
<td>Not Selected</td>
<td><img src="ConstraintLinkDiagram.png" alt="Constraint Link Diagram" /></td>
</tr>
<tr>
<td><strong>Copy Link</strong></td>
<td>Link is generated by dragging-and-dropping</td>
<td>A link between two identical schema substructures. During run time, the source data is directly copied as a block to the target data. A copy link is also generated when mapping between a untyped XML node and a typed XML complex-type node.</td>
<td>Not Selected</td>
<td><img src="CopyLinkDiagram.png" alt="Copy Link Diagram" /></td>
</tr>
</tbody>
</table>
Transforming Data Using XQuery

<table>
<thead>
<tr>
<th>Implied Link</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A link whose XQuery code can not be interpreted by the XQuery Mapper to be another type of link.</td>
<td>Example: A data link that was modified using the <strong>General Expression</strong> section of the <strong>Target Expression</strong> tab or the <strong>Source View</strong>. For example, adding the <code>fn:upper-case</code> function to a link. The data links generated between a second set of child nodes when a union constraint has been applied a set of two structural links. The child nodes must be of the same subschema.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Color of the Dashed Line is . . .</th>
<th>Means . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>No link can be created between the source node and the target node. The data type of the target node cannot be converted to the data type of the source node. (The link represents a illegal mapping.) For example, a node of data type XML string can not be converted to an XML repeating node. A red error message will be displayed when you drag the source node over the target node.</td>
</tr>
<tr>
<td>Orange</td>
<td>A link can be created between the source node and the target node but the data types are not completely compatible. An orange warning message describing the incompatibility or any necessary conversion will be displayed when you drag the source node over the target node.</td>
</tr>
<tr>
<td>Green</td>
<td>A link can be created between the source node and the target node. The data type of the target node is compatible with the data type of the source node.</td>
</tr>
</tbody>
</table>
After the source node has been dropped on the target node, a line representing a link will be displayed. Depending on the target and source nodes, a dashed line or a solid line will be displayed.

**XML Global Elements, Global Types, Local Elements, and Attributes**

An XML Schema type or element is considered *global* if it is a direct child of the `schema` element while an XML Schema type or element is considered *local* if it is not a direct child of the `schema` element (is nested to another element) as shown in the following XML Schema.

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.acme.org/globalExample"
  xmlns="http://www.acme.org/globalExample"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="globalElement">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="localElement"
          minOccurs="1" maxOccurs="1"
          type="xs:string" />
      </xs:sequence>
      <xs:attribute name="attribute"
        type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:complexType name="globalType">
    <xs:sequence>
      <xs:element name="anotherLocalElement"
        minOccurs="0" maxOccurs="unbounded"
        type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

In the preceding example XML Schema, the `globalElement` is *global* because it is a direct child of the `schema` element while `localElement` is *local* because it is child of `globalElement`. 
You can also define a global type as shown by the `globalType` element at the bottom of the preceding XML Schema. While you can only have one global element in an XML Schema, you can declare many elements (with different names) of the same global type in a single XML Schema.

The following table shows the graphical representations of these different XML components in the XQuery Mapper.

<table>
<thead>
<tr>
<th>Name</th>
<th>Representation in the XQuery Mapper</th>
<th>Name in Preceding Example XML Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Element</td>
<td>![Diagram of global element]</td>
<td><code>globalElement</code> and <code>anotherGlobalElement</code></td>
</tr>
<tr>
<td>Local Element</td>
<td>![Diagram of local element]</td>
<td><code>localElement</code></td>
</tr>
<tr>
<td>Global Type</td>
<td>![Diagram of global type]</td>
<td><code>globalType</code></td>
</tr>
<tr>
<td>Attribute</td>
<td>![Diagram of attribute]</td>
<td><code>attribute defined for globalElement</code></td>
</tr>
</tbody>
</table>
Examples: Manipulating and Constraining Data Using XQuery

This section provides example scenarios for the Sample Project located in your BEA XQuery Mapper installation. To learn more about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

This section includes the following topics:

- Combining Data From Different Schemas
- Mapping Repeating Elements and Creating a Join
- Using the Union Option of the Constraints Tab
- Creating a Transformation Between a Repeating Source and Non-Repeating Target
- Creating a Transformation Between a Non-Repeating Source and Repeating Target
- Creating a Nested If-Then-Else Expression
- Using Recursive Schemas
- Creating Group By Key Fields XQuery Constructs
- Testing XQuery Files

Combining Data From Different Schemas

You can use the XQuery Mapper to combine the contents of two different schemas, as shown in the following Figure 3-1.
In this example, customer data (as described in the CustInfo.xsd schema) is merged with a repeating element (line-items, as described in the PO.xsd schema) to form a single XML document valid against the POCustInfo.xsd schema.

Figure 3-1 Combining Data From Different Schemas

To Combine Data from Different Schemas

1. Launch Eclipse and navigate to the Sample Project.
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the XQuery Transformation folder.


4. Verify name of parent folder.
   For this example, the parent folder should be Samples/XQueryTransformations
5. In the Filename path, enter `combineData`.

6. Select the following source files:
   - CustInfo.xsd\customer
   - PO.xsd\purchase-order

7. Select the following target schema and element:
   - POCustInfo.xsd\purchase-order

   After selecting the source and target data, the `combineData.xq` file is created.

8. Select the **Source** elements described in the following table and drag them to the **Target** pane.

<table>
<thead>
<tr>
<th>Source</th>
<th>Links to...</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td></td>
<td>purchase-order\customer</td>
</tr>
<tr>
<td>customer\customer-id</td>
<td></td>
<td>purchase-order\customer\customer-id</td>
</tr>
<tr>
<td>customer\customer-name</td>
<td></td>
<td>purchase-order\customer\customer-name</td>
</tr>
<tr>
<td>customer\customer-address</td>
<td></td>
<td>purchase-order\customer\customer-address</td>
</tr>
<tr>
<td>purchase-order\line-item</td>
<td></td>
<td>purchase-order\line-items\line-item</td>
</tr>
<tr>
<td>purchase-order\line-items\line-item\part-no</td>
<td></td>
<td>purchase-order\line-items\line-item\part-no</td>
</tr>
<tr>
<td>purchase-order\line-items\line-item\quantity</td>
<td></td>
<td>purchase-order\line-items\line-item\quantity</td>
</tr>
</tbody>
</table>

As shown in the following figure, a dotted line represents a **Structural** link. This type of link is created between two parent structures that do not map data directly.

As shown in the following figure, a solid line represents a **Data** link. This type of link converts the value of the source node directly to the value of the target node.

To learn more about links, see “Understanding Design View Graphical Representations” on page 2-23.
The map between the **Source** and **Target** elements is shown in the following figure.

**Figure 3-2  Data Transformation in Design View**

For information about testing XQuery Transformations, see “Testing XQuery Files” on page 3-34.

**Mapping Repeating Elements and Creating a Join**

In this section, `PriceQuote.xsd`, `AvailableQuote.xsd`, and `taxrate.xsd` are joined to create a single `Quote.xsd` file.

This example includes the following steps:

- Step 1. Create a New XQuery File
- Step 2. Add a Conditional Constraint
- Step 3. Add Links to Populate Empty Element
- Step 4. Calculate Total Cost
- Step 5. Add a Constraint With Multiple Conditions

**Step 1. Create a New XQuery File**

In this section you will create an XQuery transformation using the `AvailQuote.xsd`, `PriceQuote.xsd`, and `taxrate.xsd` files. After doing so, you will map several `priceQuote` and `availRequest` source elements to corresponding target elements.
1. Launch Eclipse and navigate to the **Sample Project**.
   – For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   – For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the **XQuery Transformation** folder.

3. Choose **New → XQuery Transformation**.

4. Verify that the parent folder is **Samples/XQueryTransformations**

5. In the Filename path, enter **Join**.

6. Select the following **source** schema files and elements:
   - PriceQuote.xsd\priceQuote
   - PriceQuote.xsd\taxRate
   - AvailQuote.xsd\availRequest

7. Select the following **target** schema and element:
   - Quote.xsd\quote

   After selecting the source and target data, the **Join.xq** file is created in the **Samples/XQueryTransformations** folder.

8. Click the **Source** elements described in the following table and drag and drop them on the **Target** element indicated to create a link between the elements.

<table>
<thead>
<tr>
<th>Source Element</th>
<th>Links to….</th>
<th>Target Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>priceQuote\customerName</td>
<td></td>
<td>quote\name</td>
</tr>
<tr>
<td>priceQuote\shipAddress</td>
<td></td>
<td>quote\address</td>
</tr>
<tr>
<td>priceQuote\priceRequests\priceRequest</td>
<td></td>
<td>quote\quoteResponse</td>
</tr>
<tr>
<td>availRequest</td>
<td></td>
<td>quote\quoteResponse</td>
</tr>
</tbody>
</table>
9. The **Design** tab displays the map you created as shown in the following figure.

**Figure 3-3 Join Example**

![Figure 3-3 Join Example](image)

10. Save your changes.

### Step 2. Add a Conditional Constraint

The source documents `priceQuote` and `availRequest` share the common element, `widgetId`. In this sub-section, you add a conditional constraint that specifies if the `widgetId` of the `availRequest` element is equal to the `widgetId` of the `priceRequest` element, then the merged repeating element `quoteResponse` should be returned.

**To Add a Constraint**

1. Open the `join.xq` in the **Design** view.

2. In the **Source** pane, click the following element:

   `priceRequests/priceRequest/widgetId`

3. Drag-and-drop `priceRequests/priceRequest/widgetId` onto the following element in the **Source** pane:

   `availRequest/widgetId`

   A line between the two `widgetId` nodes (to indicate that you merged the elements) is displayed, as shown in the following figure.
Figure 3-4  Adding a Conditional Constraint to a Join

4. Save your changes.

5. View the changes by clicking the **Source** tab.

   The link between the widgetId nodes generates:
   
   - The *where* clause in the *for* loop.
   - This *where* clause constrains or limits the output of the *for* loop.
   - The *where* clause specifies that if the expression in the *where* clause is true, the *for* loop will output the contents of the return.
   - For this example, if the widgetId of the availRequest element is equal to the widgetId of the priceRequest element, the following XML data is returned: 

     ```xml
     </quoteResponse>
     ```

The *quoteResponse* element is empty. Use the procedure in “Step 3. Add Links to Populate Empty Element” on page 3-8 to add data links that will populate the *quoteResponse* element.
Step 3. Add Links to Populate Empty Element

Use the procedure in this section to add data links to populate the `quoteResponse` element.

**To Populate the QuoteResponse Element**

1. Open Join.xq in the **Design** view.

2. Click the **Source** elements described in the following table and drag and drop them on the **Target** element indicated to create a link between the elements.

<table>
<thead>
<tr>
<th>Source Element</th>
<th>Links to....</th>
<th>Target Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>priceQuote\priceRequests\priceRequest\price</code></td>
<td>➔</td>
<td><code>quote\quoteResponse\unitprice</code></td>
</tr>
<tr>
<td><code>priceQuote\priceRequests\priceRequest\widgetid</code></td>
<td>➔</td>
<td><code>quote\quoteResponse\widgetid</code></td>
</tr>
<tr>
<td><code>availRequest\requestedQuantity</code></td>
<td>➔</td>
<td><code>quote\quoteResponse\requestedQuantity</code></td>
</tr>
<tr>
<td><code>availRequest\quantityAvail</code></td>
<td>➔</td>
<td><code>quote\quoteResponse\fillOrder</code></td>
</tr>
<tr>
<td><code>availRequest\shipDate</code></td>
<td>➔</td>
<td><code>quote\quoteResponse\shipDate</code></td>
</tr>
</tbody>
</table>

After creating the preceding links, the following map is displayed in the **Design** tab.

**Figure 3-5 Populating the QuoteResponse Element**
Step 4. Calculate Total Cost

In this step, you will add the function for calculating the total cost of the purchase order to the XQuery source.

To Calculate Total Cost

1. Open the Join.xq file in Source view.
2. Add the following function declaration to the XQuery:

```
Listing 3-1 calculateTotalPrice Function

declare function xf:calculateTotalPrice($taxRate as xs:float, $quantity as xs:float,$price as xs:float)
    as xs:float {
    let $taxQuantity := ($taxRate * $quantity)
    let $totalTax := ($taxQuantity * $price)
    let $costNoTax := ($quantity * $price)
    let $totalCost := ($totalTax + $costNoTax)
    return $totalCost
};
```

3. Open the Join.xq file in Design view.
   
   **Note:** Your Join.xq file now includes two function declarations: calculateTotalPrice and Join. Note that when you have more than one function in an XQ file, the function with the same name as the XQ file is rendered in the Design View. (In this case, the Join function is displayed in the Design view.)

4. In the Target pane, select the totalCost node. Keep it selected for the next step.

5. Navigate to the Target Expression tab and choose General.

6. To calculate the totalCost, add the following General Expression:

```
xf:calculateTotalPrice($taxRate,$availRequest/ns1:requestedQuanity,$priceRequest/ns0:price)
```
7. Click **Apply**.
   The expression is added to the **totalCost** element in your XQuery.

8. Save your changes.
   The design view reflects the new calculation for **totalCost**.

**Step 5. Add a Constraint With Multiple Conditions**

Creating a constraint using the **Where Clause Expression** pane of the **Constraints** tab adds a `where` clause to the XQuery `for` loops—limits the target repeating elements that are returned during run time.

During run time, the `for` loop will iterate only over those repeating elements that meet the complex condition. In this section, you will add another condition (resulting in a complex condition) to the `where` of the `for` loop to further limit what is returned by the `for` loop.
To Add a Constraint with Multiple Conditions

1. Open the join.xq file.
2. Select the link between:
   
   availRequest
   and
   quote\quoteResponse

   The single condition that makes up the where clause is displayed in Where Clause Expression pane of the Constraints tab.

   data($priceRequest/ns0:widgetId) = data($availRequest/ns1:widgetId)

3. Select the availRequest/requestedQuanity node and drag-and-drop it into the Left Hand Expression section of the Where Clause Expression pane to create:

   data($availRequest/ns1:requestedQuanity)

4. Select the < operator.
5. Remove the text in the Right Hand Expression section of the Where Clause Expression pane.
6. In the Right Hand Expression section of the Where Clause Expression pane, enter “50”.
   
   Note: You must use the quotes around 50. That is, enter “50”, not 50.
7. From the Join Type field select the AND option.
   
   The Join Type determines how the conditions that make up where clause are evaluated during run time.
8. Click Add. The second condition is added to the where clause of the for loop.
9. Save your work. This step completes the creation of the following where clause:

   where (data($availRequest/ns1:widgetId) =
   data($priceRequest/ns0:widgetId)
   and data($availRequest/ns1:requestedQuanity) < "50")
Examples: Manipulating and Constraining Data Using XQuery

**To Test the XQuery**

For information about testing XQuery transformations, see “Testing XQuery Files” on page 3-34.

Specifically for this example, complete the following steps to set up the test to test that the XQuery works when both the constraints you configured in “Step 5. Add a Constraint With Multiple Conditions” on page 3-10 are met:

1. In the **Source Data** pane in the Test view, select **priceQuote** in the **Source variable** field and click Generate Data.

2. Note the value of the widgetId element in the test XML.
   
   `<ns0:widgetId>value</ns0:widgetId>`

3. In the **Source Data** pane, select **availRequest** in the **Source variable** field and click Generate Data.

4. Note the value of the widgetId element in the test XML. Edit the value to match the value displayed in the **priceQuote** test XML file (step 2).
   
   `<ns0:widgetId>value</ns0:widgetId>`

5. Locate the **requestedQuanity** element and edit the value to a value less than 50. For example:
   
   `<ns0:requestedQuanity>25</ns0:requestedQuanity>`

6. In the **Result Data** pane, click **Test XQuery** and view the results of the XQuery in the **Result Data** pane.

**Using the Union Option of the Constraints Tab**

In this example, you will use the **Union** option in the **Constraints** tab to construct an XQuery that maps data of the same type into larger sets of data.

**To Combine Sets of Data of the Same Type**

1. Launch Eclipse and navigate to the **Sample Project**.
   
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the **XQuery Transformation** folder.

4. Verify the parent folder.
   In this example the parent folder is Samples/XQueryTransformations.

5. In the Filename path, enter union.

6. Select the following source file (select PO.xsd\purchase-order twice):
   PO.xsd\purchase-order
   PO.xsd\purchase-order

7. Select the following target schema and element:
   Order.xsd\order
   After selecting the source and target data, the union.xq file is created.

8. Select the Source elements described in the following table and drag them to the Target pane.

<table>
<thead>
<tr>
<th>Source Element</th>
<th>Links to...</th>
<th>Target Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>purchase-order\line-items\line-item</td>
<td></td>
<td>order\items\item</td>
</tr>
<tr>
<td>purchase-order\line-items\line-item</td>
<td></td>
<td>order\items\item</td>
</tr>
</tbody>
</table>

   The following graphic displays the Design tab of the linking.

   **Figure 3-8 Creating a Union**

9. Verify that the link between the $purchase-order/line-items/line-item (repeating element in the Source pane) and the order/items/item (repeating element in the Target pane) is selected before you proceed.
10. In the **Constraint Type** pane, select **Union**.

11. Create a link between the **part-no** element in the **Source** pane and the **part-number** element in the **Target** pane.

*Note:* In the **Source** and **Target** pane, if the element names are the same, you can use the **Induce Map** right-click menu option instead of manually creating the link as described in this step. For more information, see “**Link Menu Options**” on page 2-24.

Because the two structural links have the union constraint applied to them, a set of implied data links between the second set of sub-elements is generated as shown in Figure 3-9. The solid gray lines represents implied links that were created by selecting **union** in the **Constraints** Tab.

**Figure 3-9 Creating Implied Links**

12. Save your work.

For more information about testing XQuery transformations see “**Testing XQuery Files**” on page 3-34.

**Creating a Transformation Between a Repeating Source and Non-Repeating Target**

This example shows how to map a repeating element a non-repeating single element. In this example, you will create a transformation that takes the value of a repeating XML element (defined by a source schema) and maps it to a single element in a target XML document (defined in a target schema) as shown in the following figure.
Creating a Transformation Between a Repeating Source and Non-Repeating Target

Transforming Data Using the XQuery Mapper

To Create a Map Between a Repeating Source Group and a Non-Repeating Target Element

1. Launch Eclipse and navigate to the Sample Project.
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the XQuery Transformation folder.


4. Verify the parent folder.
   In this example the parent folder is Samples/XQueryTransformations

5. In the Filename field, enter repeatToNonRepeat.

6. Select the following source file:
   Dates.xsd\dates

---

Figure 3-10 Repeating Source Group to Non-Repeating Target Element

Source XML Data (valid to the XML Schema in Dates.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<dates xmlns="http://www.acme.org/Dates">
  <date>
    <type>DELIVERY</type>
    <value>03/31/2004</value>
  </date>
  <date>
    <type>BILLING</type>
    <value>03/29/2004</value>
  </date>
</dates>
```

Result XML Data (valid to the XML Schema in PODate.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<po:PODate xmlns:po="http://www.acme.org/PODate">
  <po:billing-date>03/29/2004</po:billing-date>
  <po:delivery-date>03/31/2004</po:delivery-date>
</po:PODate>
```

During run time, the first generated XQuery for loop iterates through the list of source repeating elements of type date. If the constraint (date/type = "BILLING") is true, then the contents of the date/value element is returned as the contents of the target PO Date/billing-date element.

During run time, the second generated XQuery for loop iterates through the list of source repeating elements of type date. If the constraint (date/type = "DELIVERY") is true, then the contents of the date/value element is returned as the contents of the target PO Date/delivery-date element.
Examples: Manipulating and Constraining Data Using XQuery

7. Select the following target schema and element:
   PODate.xsd\PODate

8. In the Source pane, select the repeating element (dates/date) and drag it the first single element (PODate/billing-date) in the Target pane.
   Keep this link selected for the next step.

9. Select the Constraints tab.

10. From the Source pane, select the dates/date/type node and drop it into the Left Hand Expression pane of the Where Clause Expression in the Constraints tab.

11. Select the operator: =

12. In the Right Hand Expression pane of the Where Clause Expression in the Constraints tab, enter the string "BILLING".  
    Use quotes when entering a date string, as shown in the following figure.

Figure 3-11  Use Quotes when Entering String Functions

13. Click Add.

   The constraint created in the preceding steps specifies that the value of the dates/date/type element in an XML document is compared to the value "BILLING".

   In the next steps, you will add XQuery code to the for loop to return specified data when the value of the dates/date/type element equals "BILLING".

14. In the Source pane, select the dates/date/value element and drag to the PODate/billing-date element in the Target pane.

   A data link is created.

   During run time, this data link will return the value of datesDoc/date/value as the value of billing-date if the constraint: data($date/ns0:type) = "BILLING" evaluates to true.
15. In the **Source** pane, select the repeating element (dates/date) and drag it the second single element (PODate/delivery-date) in the **Target** pane.

A dashed line linking the two elements is displayed.

Keep this link selected for the next step.

16. From the **Source** pane select the dates/date/type node and drop it into the **Left Hand Expression** pane of the **Where Clause Expression** in the **Constraints** tab.

17. Select the operator: =

18. In the **Right Hand Expression** pane of the **Where Clause Expression** in the **Constraints** tab, enter the string "DELIVERY". The next step will add the XQuery code to return data if during run time the constraint is equal to true.

**Note:** Use quotes when entering string values.

19. Click **Add**.

During run time, the constraint created in step, tests if datesDoc/date/type is equal to the string: "DELIVERY".

20. In the **Source** pane, select the datesDoc/date/value element and drag it the PODate/delivery-date element in the **Target** pane.

A data link is created.

During run time, this data link will return the value of dates/date/value as the delivery-date if the constraint: data($date/ns0:type) = "DELIVERY" evaluates to true.

21. Save your work.

For more information about testing XQuery files, see “Testing XQuery Files” on page 3-34.

---

**Creating a Transformation Between a Non-Repeating Source and Repeating Target**

This example shows how to map a a non-repeating element to a a repeating element. In this example, you will create a transformation that during run time will take a single source element and maps it to repeating target element as shown in the following figure.
Figure 3-12 Non-Repeating Source Element to Repeating Target Group

During run time, the query executes the following actions:

1. If the name of the source input element is equal to "billing-date", then the string "billing" is returned as the data for the value node.

2. The value of the source XML element billing-date becomes the value of the target type XML element.

3. If the name of the source input element is equal to "delivery-date", then the string "DELIVERY" is returned as the data for the value node.

4. The value of the source XML element delivery-date becomes the value of the target type XML element.

To create a Non-Repeating Source Element and a Repeating Target Group

1. Launch Eclipse and navigate to the Sample Project.
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the XQuery Transformation folder.


4. Verify the parent folder.

   In this example the parent folder is Samples/XQueryTransformations.

5. In the Filename path, enter nonRepeatToRepeat.
6. Select the following source file:
   PODate.xsd\PODate

7. Select the following target schema and element:
   Dates.xsd\dates
   After selecting the source and target data, the nonRepeatToRepeat.xq file is created.

8. Select the Source elements described in the following table. Drag them from the Source pane to the Target pane.

<table>
<thead>
<tr>
<th>Source Pane</th>
<th>Links to....</th>
<th>Target Pane</th>
</tr>
</thead>
<tbody>
<tr>
<td>pODate/billing-date</td>
<td></td>
<td>dates/date</td>
</tr>
<tr>
<td>pODate/delivery-date</td>
<td></td>
<td>dates/date</td>
</tr>
</tbody>
</table>

   The following XQuery code is generated:
   
   ```xml
   <ns1:dates>
   { for $PODate in $PODateDoc/ns0:billing-date union $PODateDoc/ns0:delivery-date
     return <ns1:date/>
   }
   </ns1:dates>
   
   During run time, the for loop in the preceding XQuery code is executed twice. The first time the for loop is run, the iteration variable $PODate is equal to the first element in the union: $PODateDoc/ns0:billing-date and the second time the for loop is run the iteration variable $PODate is equal to the second element in the union: $PODateDoc/ns0:delivery-date.

   The XML data returned by the preceding query returns two empty elements:
   <ns1:date/>.

   The following steps will add the XQuery code to return the billing and delivery dates to the query.

9. In the Source pane, select the pODate/billing-date element and drag it the dates/value element in the Target pane.

   Two data links are created as shown in the following figure.
The structural links (pODate/billing-date to dates/date) and
(pODate/delivery-date dates/date) are joined when you created the link from the
pODate/billing-date element to the dates/value element (dates/value), a second
data link between the pODate/delivery-date element and dates/value element is
automatically created.

10. In the Source pane, select the pODate/billing-date element and drag it the dates/type
element in the Target pane.

Two data links are created.

Keep the pODate/billing-date to dates/type link selected for the next step.

11. Select the Target Expression tab.

12. Select the If Then Else option.

The XQuery if-then-else construct is added to the link. For example, the following XQuery
source code segment for the link is replaced:

```xquery
data($PODate)
```

By the following XQuery source code segment for the link:

```xquery
if (xf:boolean("true")) then
data($PODate)
else
()```

13. Click the If Condition.

The If Condition pane is displayed.

14. In this step, you add a condition to the if section of the if-then-else:

a. Select the Expression Functions view.

b. Expand XQuery Functions→Node Functions.
Creating a Transformation Between a Non-Repeating Source and Repeating Target

c. Select the `local-name` function and drag it into the Left Hand Expression pane of the If Condition pane. Leave the $node-var argument selected.

d. Select the Expression Variables view.

e. From the Source, drag-and-drop the `pODate` Structural Link variable over the $node-var argument of the local-name function in the If Condition pane.

f. Select the operator: =

g. In the Right Hand Expression section of the If Condition pane, enter “BILLING DATE” then click Add.

   **Note:** Use quotes when entering string values.

As shown in the following figure, the condition is added to the if section of the if-then-else.

![Figure 3-14 If Condition for Billing Date](image)

15. Click the Then Expression.

16. In the Then Expression section, replace the existing text with the string: “BILLING”

Replace the following text:

```
data ($pODate)
```

With the following string:

```
“BILLING”
```

   **Note:** You must use quotes around the “BILLING” string.

17. Click Apply.

18. Click the Else Expression.

19. In the Else Expression section, enter the string “DELIVERY”

   **Note:** You must enter quotes around the “DELIVERY” string.
20. Click **Apply**.

21. In the **Expression Structure** pane, click **If Then Else**.

   The XQuery code is displayed in the **Expression Structure** pane, as shown in the following figure.

   **Figure 3-15 Viewing the If-Then-Else Sample Results**

```
if (fn:local-name($sp0Data) = "BILLING DATE") then
  "BILLING"
else
  "DELIVERY"
```

22. Save your work.

For more information about testing XQuery files, see “Testing XQuery Files” on page 3-34.

**Creating a Nested If-Then-Else Expression**

In this example, you will be creating an XQuery transformation that calculates price based on a widget ID and state tax rate. Using the XQuery Mapper, you will create an expression structure that represents the following *if-then-else* logic:

- *If* the widget ID is between 0-200, *then* the price is $10.00
- *Else if* the widget ID is between 201-400, *then* price is $20.00
- *Else If* the widget ID is between 401-600, *then* price is $30.00

This example includes the following steps:

- **Step 1. Create New XQuery Transformation**
- **Step 2. Create First If Condition**
- **Step 3. Create First Nested If-Then-Else Condition**
- **Step 4. Create Second Nested If-Then-Else Condition**
Step 1. Create New XQuery Transformation

In this step, you will create a new XQuery transformation using the PurchaseAgree.xsd and Supplier.xsd.

To Create a New XQuery Transformation

1. Launch Eclipse and navigate to the Sample Project.
   – For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   – For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.
2. Right-click the XQuery Transformation folder.
4. Verify the parent folder.
   In this example the parent folder is Samples/XQueryTransformations.
5. In the Filename path, enter ifthenelse.
6. Select the following source file:
   Supplier.xsd\Supplier
7. Select the following target schema and element:
   PurchaseAgree.xsd\PurchaseOrder
   After selecting the source and target data, the ifThenElse.xq file is created.
8. Drag the following elements from the Source pane to the Target pane.

<table>
<thead>
<tr>
<th>Source Pane</th>
<th>Links to...</th>
<th>Target Pane</th>
</tr>
</thead>
<tbody>
<tr>
<td>supplier/products/product</td>
<td>→</td>
<td>PurchaseOrder/products/product</td>
</tr>
<tr>
<td>supplier/products/product/price</td>
<td>→</td>
<td>PurchaseOrder/products/product/price</td>
</tr>
</tbody>
</table>
Step 2. Create First If Condition

In this step, you will be creating an *If* expression that states: *If* the widget ID is between 0-200, *then* the price is $10.00.

**To Create the First If Expression**

1. Click the **Target Expression** Tab.
2. Choose the Expression Type **If Then Else**.
   
   In the Expression Structure pane, verify that the **If Condition** is highlighted.
3. From the **Source** pane, drag and drop the **widgetID** element to the **Left Hand Expression** pane.
4. Select the operator: >
5. In the **Right Hand Expression** pane, enter “0”.
   
   **Note:** Use quotes when entering a numeric value.
6. Click **Add**.
7. Select the operator: <=
8. In the **Right Hand Expression** pane, enter “200”.
   
   **Note:** Use quotes when entering a numeric value.
9. Click **Add**.
10. Click the **Then Expression**.
11. In the **Edit Then Expression** pane, enter “$10.00”.
    
    **Note:** Delete existing data in the **Edit Then Expression** pane. Use quotes when entering “$10.00”.
12. Click **Apply**.
13. In the **Expression Structure** pane highlight **If Then Else**.
    
    Validate that your *if then* statement appears as shown in the following:
Creating a Nested If-Then-Else Expression

Step 3. Create First Nested If-Then-Else Condition

In this step, you will be creating a nested *If* expression that states: *If* the widget ID is between 201-400, *then* the price is $20.00. To accomplish this, you will insert a nested *If-Then-Else* inside the *Else Expression* created in “Step 2. Create First If Condition” on page 3-24.

To Create the First Nested If-Then-Else Condition

1. In the **Expression Structure** pane select, **Else Expression**.
2. Right-click and choose **Insert Nested If-Then-Else**.
3. In the **Nested If-Then-Else** expression, highlight the **If Condition**.
4. From the **Source** pane, drag and drop the **widgetID** element to the **Left Hand Expression** pane.
5. Select the operator: >
6. In the **Right Hand Expression** pane, enter “201”.
   **Note:** Use quotes when entering a numeric value.
7. Click **Add**.
8. Select the operator: <=
9. In the **Right Hand Expression** pane, enter “400”.
   **Note:** Use quotes when entering a numeric value.
10. Click **Add**.
11. Click the **Then Expression**.
12. In the **Edit Then Expression** pane, enter “$20.00”.
   
   **Note:** Use quotes when entering a numeric value.

13. Click **Apply**.

14. In the **Expression Structure** pane highlight **If Then Else**.
   
   Validate that your statement appears as shown in the following:

   **Figure 3-17 Nested If Then Else Expression Structure**

   ![Nested If Then Else Expression Structure](image)

   **Step 4. Create Second Nested If-Then-Else Condition**

   In this step, you will be creating an *If* expression that states: *If* the widget ID is between 401-600, *then* the price is $30.00. To accomplish this, you will insert a nested *If-Then-Else* inside the *Then Expression* created in “Step 3. Create First Nested If-Then-Else Condition” on page 3-25.

   **To Create the Second Nested If-Then-Else Condition**

   1. In the **Expression Structure** pane select, Else Expression.
      
      **Note:** Be sure to select the Else Expression created in “Step 3. Create First Nested If-Then-Else Condition” on page 3-25

   2. Right-click and choose Insert Nested If-Then-Else.

   3. From the **Source** pane, drag and drop the widgetID element to the Left Hand Expression pane.

   4. In the **Nested If-Then-Else** expression, highlight the If Condition.

   5. Select the operator: >
6. In the **Right Hand Expression** pane, enter "401".
   **Note:** Use quotes when entering a numeric value.

7. Click **Add**.

8. Select the operator: <=

9. In the **Right Hand Expression** pane, enter "600".
   **Note:** Use quotes when entering a numeric value.

10. Click **Add**.

11. Click the **Then Expression**.

12. In the **Edit Then Expression** pane, enter “$30.00”.
    **Note:** Use quotes when entering a numeric value.

13. Click **Apply**.

14. In the **Expression Structure** pane highlight **If Then Else**.
    Validate that your statement appears as shown in the following figure.

**Figure 3-18 Second Nested If-Then-Else Expression**

```xml
if ($data[ns:ProdId] < "0"
and $data[ns:ns0:VendorId] <="300") then
"$30.00"
else
  if ($data[ns:ProdId] < "401"
  and $data[ns:ns0:VendorId] <="400") then
  "$20.00"
  else
    if ($data[ns:ProdId] < "901"
    and $data[ns:ns0:VendorId] <="600") then
    "$30.00"
    else
      ()
```
Using Recursive Schemas

This example shows how to create a map with schemas that have recursive elements. A recursive element contains a child element of the same type as the parent as shown in the following figure. In this example, the `product` element is a recursive element because it is of type `productType` and `productType` contains a child-product element which is also of type `productType` (`productType` refers to itself).

**Figure 3-19 Recursive Schemas Example**

```xml
 <?xml version='1.0'?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' targetNamespace='http://www.eae.org/Product' xmlns='http://www.eae.org/Product' elementFormDefault='qualified' attributeFormDefault='unqualified'>
  <xs:complexType name='productType'>
    <xs:sequence>
      <xs:element name='Product'>
        <xs:element name='part-description' minOccurs='0' maxOccurs='unbounded' type='xs:string' />
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

**To Create a Transformation Using Recursive Schemas**

1. Launch Eclipse and navigate to the Sample Project.
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the XQuery Transformation folder.


4. Verify the parent folder.
   In this example the parent folder is, Samples/XQueryTransformations.

5. In the Filename path, enter recursive.
6. Select the following source file:
   SupplierAcme.xsd\supplier-acme

7. Select the following target schema and element:
   Product.xsd\product
   After selecting the source and target data, the recursive.xq file is created.

8. Select the Source elements described in the following table. Drag them from the Source pane to the Target pane.

<table>
<thead>
<tr>
<th>Source Pane</th>
<th>Links to....</th>
<th>Target Pane</th>
</tr>
</thead>
<tbody>
<tr>
<td>supplier-acme\part-description-00100</td>
<td>→ product\part-description</td>
<td></td>
</tr>
<tr>
<td>supplier-acme\part-description-00101</td>
<td>→ product\child-product\part-description</td>
<td></td>
</tr>
<tr>
<td>supplier-acme\part-description-00101</td>
<td>→ product\child-product\child-product\part-description</td>
<td></td>
</tr>
</tbody>
</table>

As shown in the following figure, the product element contains the recursive child-product element.

**Figure 3-20 Mapping Recursive Elements**

9. Save your work.

For more information about testing XQuery files, see “Testing XQuery Files” on page 3-34.
Creating Group By Key Fields XQuery Constructs

You can use the **Group by Key Fields** functionality to group data based on one or more key values. However, the **Group By** functionality is not supported graphically in the XQuery Mapper and there is no representation of the XQuery in the Mapper’s Design View. Therefore you must write the **Group By** XQuery construct in the XQuery Mapper’s Source View.

In this example, the input-warehouse-id and the input-location-desc elements are used as the key fields to group the data in the output document:

- The first and third instances of the input-line-item repeating element in the input XML document (see Listing 3-2) contain the same values for the input-warehouse-id element and the input-location-desc elements. (The values are Warehouse1 and Location1 respectively.)

- As a result of the XQuery described in this example, the first and third instances of the line items are grouped. That is, these elements are grouped with the Warehouse1 and Location1 keys in the target (output) document (see Listing 3-3).

Listing 3-2  Example Input XML Document

```xml
<input-warehouse-inventory xmlns="http://www.creditpo.org/repkeyin">
  <input-line-item>
    <input-warehouse-id>Warehouse1</input-warehouse-id>
    <input-location-desc>Location1</input-location-desc>
    <input-part-no>1</input-part-no>
    <input-quantity>10</input-quantity>
  </input-line-item>
  <input-line-item>
    <input-warehouse-id>Warehouse2</input-warehouse-id>
    <input-location-desc>Location2</input-location-desc>
    <input-part-no>2</input-part-no>
    <input-quantity>20</input-quantity>
  </input-line-item>
</input-warehouse-inventory>
```
Listing 3-3  Example Output Document

<ns0:output-inventory xmlns:ns0="http://www.creditpo.org/repkeyout">  
  <ns0:output-warehouse-inventory>
    <ns0:output-warehouse-id>Warehouse1</ns0:output-warehouse-id>
    <ns0:output-location-desc>Location1</ns0:output-location-desc>
    <ns0:output-line-item>
      <ns0:output-part-no>1</ns0:output-part-no>
      <ns0:output-quantity>10</ns0:output-quantity>
    </ns0:output-line-item>
    <ns0:output-line-item>
      <ns0:output-part-no>3</ns0:output-part-no>
      <ns0:output-quantity>30</ns0:output-quantity>
    </ns0:output-line-item>
  </ns0:output-warehouse-inventory>
  <ns0:output-warehouse-inventory>
    <ns0:output-warehouse-id>Warehouse2</ns0:output-warehouse-id>
    <ns0:output-location-desc>Location2</ns0:output-location-desc>
    <ns0:output-line-item>
      <ns0:output-part-no>3</ns0:output-part-no>
      <ns0:output-quantity>30</ns0:output-quantity>
    </ns0:output-line-item>
  </ns0:output-warehouse-inventory>
</ns0:output-inventory>
To Create a Group By XQuery Construct

1. Launch Eclipse and navigate to the Sample Project.
   - For more information about launching Eclipse, see “Launching the XQuery Mapper” on page 2-2.
   - For more information about opening the Sample Project, see “XQuery Mapper Sample Project” on page 2-3.

2. Right-click the XQuery Transformation folder.


4. Verify the parent folder is Samples/XQueryTransformations.

5. In the Filename path, enter groupby.

6. Select the following source file and element:
   regroupKeyFldIn.xsd\input-warehouse-inventory

7. Select the following target schema and element:
   regroupKeyFldOut.xsd\output-inventory

   After selecting the source and target data, the groupby.xq file is created.

   **Note:** At this point, if you were working in the XQuery Mapper in the BEA WebLogic Integration 8.1 release, you could highlight the link between input-warehouse-id and output-warehouse-inventory and choose the Constraint Type Group by Key Fields graphically. However, the Design View representation for this is not available in this release. However, the Group by type is supported—you must however write the XQuery for it as described in the following steps.

8. Click the Source tab.
9. Replace the existing XQuery code with the following code.

Listing 1  XQuery for Group By Key Construct

```xml
declare namespace ns0 = "http://www.creditpo.org/repkeyin";
declare namespace ns1 = "http://www.creditpo.org/repkeyout";

declare function Regrouping($input-warehouse-inventory as element(ns0:input-warehouse-inventory))
    as element(ns1:output-inventory) {
    <ns1:output-inventory>
    {
        for $input-line-item in $input-warehouse-inventory/ns0:input-line-item
            group $input-line-item as $group by
            $input-line-item/ns0:input-warehouse-id as $key0,
            $input-line-item/ns0:input-location-desc as $key1
            return
        <ns1:output-warehouse-inventory>
        <ns1:output-warehouse-id>{ data($key0) }</ns1:output-warehouse-id>
        <ns1:output-location-desc>{ data($key1) }</ns1:output-location-desc>
        </ns1:output-warehouse-inventory>
    for $group0 in $group return
        <ns1:output-line-item>
        <ns1:output-part-no>{ xs:byte( data($group0/ns0:input-part-no) ) }</ns1:output-part-no>
        <ns1:output-quantity>{ xs:byte( data($group0/ns0:input-quantity) ) }</ns1:output-quantity>
    </ns1:output-line-item>
    }
    </ns1:output-inventory>
```
Examples: Manipulating and Constraining Data Using XQuery

```
</ns1:output-warehouse-inventory>
}
</ns1:output-inventory>
};
declare variable $input-warehouse-inventory as
element(ns0:input-warehouse-inventory) external;
Regrouping($input-warehouse-inventory)
```

10. Save your work.

The changes are not visible in the **Design** view. Use the following procedure to **Test** the XQuery.

**Testing the Group By Key Fields XQuery Construct**

1. With the `groupby.xq` file open in the Source view, click the **Test** tab.
2. In the **Source Data** tab, click **Import** ...
3. Import the `Regrouping.xml` file provided in the Sample project. That is, import `Samples/XML/Regrouping.xml`.

In the **Result Data** pane, click **Test XQuery.** The resulting data is displayed in the Result Data tab. It displays the line items grouped by key fields similar to that shown in Listing 3-3 (`input-warehouse-id` is the first key and `input-location-desc` is the second).

**Testing XQuery Files**

After creating an XQuery transformation in the **Design** tab, you can test it using the **Test** tab.

When testing XQuery transformations, you can see if the expected XML or Non-XML output is properly generated. You can use the *auto-generated* XML files or you can use your own *custom* XML and Non-XML testing files. For more information about XQuery testing, see “Testing Map Transformations” on page 2-21.

This sections contains the following topics:

- Using Auto-Generated XML Files for Testing
- Using Custom XML Files for Testing
- Using Custom MFL (Non-XML) Files for Testing
Testing XQuery Files

Using Auto-Generated XML Files for Testing

The XQuery Mapper automatically generates XML files for testing purposes. Non-XML test files are not automatically generated. For more information, see “Using Custom MFL (Non-XML) Files for Testing” on page 3-36.

1. Open the XQuery file for which you are testing.

2. Click the Test tab.

   A test XML file becomes available in the Source Variable drop-down menu. The test XML file is based on the source schema. This file is not automatically saved, and therefore must be saved manually using the Export icon.

3. In the Source Variable drop-down menu, select the source test XML file.

4. Optionally, in the Source Data pane, you can select from the following options:

   - **Generate Data**: Select to refresh or regenerate the auto-generated XML file.
   - **Import**: Select if you are importing custom test XML files (not using the auto-generated XML files.) For more information, see “Using Custom XML Files for Testing” on page 3-36.
   - **Export**: Select to save the auto-generated test XML data.
   - **Auto Validate**: Select to have source test XML files validated against source schema. The Auto Validate option can be permanently turned on, or you can select it on a case-by-case basis.

   **Note**: You can also manually edit the test XML by clicking the Source tab.

5. In the Result Data pane, click Test XQuery.

   The XQuery runs with the source data to generate the results. The results are displayed in the Result Data pane. If an error has occurred, you will see an error message that indicates the problem.

6. Optionally, in the Result Data pane, you can select from the following options:

   - **Export**: Select to save the test XML data.
   - **Validate**: Validates test XML against target schema. If expected element or attribute is missing, then an error will appear.
Using Custom XML Files for Testing

If the auto-generated XML files do not meet your business requirements, you can import custom XML files.

1. Open the XQuery file for which you are testing.
2. Click the Test tab.
3. In the Source Data pane, choose Import.
   
   The Import XML File dialog box opens. Use this to locate the XML files used for testing purposes. After selecting the appropriate file it becomes visible in the Source Data pane.
   
   You can also cut and paste test data into the Source Data pane.
4. After importing the test file, you have the following options:
   
   – Generate Data: Select to refresh or regenerate the XML file.
   – Export: Select to save.
   – Auto Validate: Select to have source test XML files validated against source schema. The Auto Validate option can be permanently turned on, or it can be selected on a case-by-case basis.
5. In the Result Data pane, select Test XQuery.
   
   The Source XML is tested against the XQuery and the results are displayed.
6. Optionally, in the Result Data pane, you can select from the following options:
   
   – Export: Select to re-save the imported test XML data.
   – Validate: Validates the test XML against the target schema. If an expected element or attribute is missing, then an error is displayed.

Using Custom MFL (Non-XML) Files for Testing

The XQuery Mapper does not automatically generate Non-XML files. Therefore, when testing Non-XML schema, you must import custom Non-XML test data.

1. Open the XQuery file you want to test.
2. Click the Test tab.
3. In the **Source Data** pane, choose **Import**.

   The **Import XML File** dialog box is displayed. Use this to locate Non-XML files used for testing.

   **Note:** The **Files of type** drop-down list must be changed to *.*

4. After importing the test file, you have the following options:

   - **Import**: Select to import another MFL test file.
   - **Export**: Select to save the MFL test data.

5. In the **Result Data** pane, select **Test XQuery**.

   The Source MFL is tested against the target MFL and the results are displayed.

6. Optionally, in the **Result Data** pane, you can select the following:

   **Export**: Select to save the result data.