Getting started with WebSphere Transformation Extender Design Studio on DataPower

Robert Vila

December 19, 2007

This article shows you how to get started using the WebSphere Transformation Extender Design Studio with WebSphere DataPower SOA Appliances. The article describes WebSphere Transformation Extender and its integration with DataPower, and uses examples to show you how to design transformation maps for the DataPower Appliance.

Introduction

IBM® WebSphere® Transformation Extender is IBM's universal transformation engine. In a graphical environment called the WebSphere Transformation Extender Design Studio, you can easily describe your data and map it between multiple inputs and outputs. Once the maps are defined, you can run them in a variety of different methods, including batch mode, API's are available in a variety of languages, via a Web Service call, on z/OS, and in many other ways. Additionally, the WebSphere Transformation Extender runtime engine is tightly integrated with a variety of other IBM products, such as WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSphere Process Server.

WebSphere DataPower SOA Appliances provide an array of functionality in purpose-built and easily deployable devices. Several versions of the DataPower Appliances provide different levels of functionality. DataPower XML Accelerator XA 35 provides XML, schemas, XPath, and XSLT processing. DataPower XML Security Gateway XS40 adds firewall, encryption, and access control. DataPower Integration Appliance XI50 further adds ESB functionality, including message routing and transformation.

Using WebSphere Transformation Extender Design Studio to develop transformations for DataPower Appliances is a natural extension of the platform. DataPower Appliances support high-performance XML-to-XML transformation on the device using XSLT. WebSphere Transformation Extender provides support for XML-to-Binary or Binary-to-XML transformation on the appliance without having to write any XSLT. As in the standalone product, writing transformations is as simple as using the Design Studio graphical user interface to define inputs and outputs and then map between them. This article will show you how to create several maps in the WebSphere Transformation Extender Design Studio and then run them on a DataPower Appliance.
Prerequisites
To follow the samples included with this article, you need to have the following products installed:

- WebSphere Transformation Extender Design Studio V8.1.0.2 or later
- WebSphere DataPower XI50 with updated firmware

See the product documentation for information on operating system and hardware requirements.

WebSphere Transformation Extender Design Studio
Creating transformations using WebSphere Transformation Extender Design Studio is a simple process. A primary benefits of using this product is that understanding the process of creating transformations in the standalone version of WebSphere Transformation Extender lets you leverage those transformations to create transformations targeted for DataPower.

Creating type trees
The first step in creating a transformation is to define a type tree, which is a WebSphere Transformation Extender artifact that is the data dictionary for what you want to transform. It is a schema-like representation of the data that enables WebSphere Transformation Extender to perform validation on inputs and outputs.

There are various ways to create type trees depending on what your data will be. WebSphere Transformation Extender provides a set of importers that automate this process, as explained below. These importers include XML schemas, DTDs, COBOL copybooks, CORBA IDL Files, and others. If you want to define your input data yourself, you can use the Type Designer graphical tool, which lets you specify what fields your data will contain, how it is delimited, and any type validation you want to perform.

Type trees are not tied to an input or output direction and can be reused extensively. They simply tell WebSphere Transformation Extender what type of data to expect, to ensure that everything you transform is valid.

Creating maps
Once you have defined all of the type trees you will be transforming, it’s time to start mapping. WebSphere Transformation Extender Design Studio provides a graphical tool called Map Designer that makes this process simple. First, you create a map source file with a .mms extension, which is a physical artifact that contains all the mappings that you might perform. Next you create an individual map -- map source files can contain many maps, but for the purposes of this article we will limit it to one.

Each map has input and output cards. WebSphere Transformation Extender allows multiple inputs and outputs, so you can read from various sources and write out to various places within the context of a single map. When creating an input card, the most important setting is the card name. After you define your input and output cards, mapping is as easy as dragging from an input to an output field. To perform more complex functions, WebSphere Transformation Extender provides built-in functions for tasks such as converting your data, performing mathematical operations on it,
or manipulating strings. When you have mapped all the output fields, you are ready to build your map and run it, which you do from within the Map Designer tool.

**Building and running WebSphere Transformation Extender maps on DataPower**

Now that you understand the concepts involved in creating WebSphere Transformation Extender type trees and maps, you can run some actual maps. Start with a simple "Hello World" map and then progress to a more realistic example.

**Running a "Hello World" map on DataPower**

Start by loading and running a simple WebSphere Transformation Extender map, first using the WebSphere Transformation Extender engine and then moving to DataPower:

1. Start up the WebSphere Transformation Extender Map Designer.
2. From the opening dialog, select **Open an Existing Map Source File**.
3. Browse to your WebSphere Transformation Extender install directory and then to the examples\DataPower directory, and open Mail.mms.
4. Notice that the ContactToLabel map has a DataPower icon next to it. This map is set by default to run with the DataPower engine. To run this map with the standard WebSphere Transformation Extender runtime engine, you need to change the map settings. Select **Map => Settings**. The final item in the list should be **MapRuntime**. Change it to **WebSphere Transformation Extender**.
5. As you can see, this map has a ContactFile input object and is mapping to a Label output. Expand the ContactFile structure and notice that it contains a number of contact records, each one made up of input data such as Last Name and First Name. The output structure is a Label that uses some of the input fields to create a mailing label. Some of these fields, such as FullName, are combinations of different pieces of input data.
6. Since this map is complete, you can build it and run it to verify that it works. Select **Map => Build** and then **Map => Run**.
7. To view the run results, select **Map => Run results** and then click **OK**. Two text files should appear and show that 11 input objects were processed and 5 output objects written.

You successfully ran the sample map using the WebSphere Transformation Extender engine. Setting up this map to run on DataPower is very simple.

1. Point the Map Designer to the DataPower Appliance where you will be running the map: Select **Tools => Options**.
2. Select **DataPower Maps**.
3. Set the Host and Port number of your DataPower Appliance, which will transmit data from the Map Designer to your appliance using HTTP. If you need to use Secure HTTP, check that box and set up the certificate that you are using. In this dialog, you can also make DataPower your default runtime engine and save the messages that Map Designer exchanges with DataPower. You can simply leave these unchecked for now.
4. Now that you have pointed to DataPower, change the map back to the DataPower runtime engine. Select **Map => Settings** and change the Map runtime back to **WebSphere DataPower**.

5. Now you can build and run the map just like before. Select **Map => Build** and then **Map => Run**. The Map Designer sends all files needed for transformation, including the data, to the DataPower engine, which transforms the data and sends the results back to Map Designer.

6. To view the run results, select **Map => Run results**.

You have now taken a simple map and shown how easy it is to run it on both the WebSphere Transformation Extender runtime engine and the DataPower runtime engine. The next step is a more real-world example.

### Creating an XML-to-COBOL map

One of the most powerful use cases for using the WebSphere Transformation Extender Design Studio with DataPower is connecting your legacy applications with new services that can consume XML. The Design Studio makes this simple and lets you retain the benefits of using the DataPower Appliance. In this example you will read in XML data and write it out in COBOL copybook format. The data involves a simple purchase order. Here is the schema:

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="item" type="xsd:string"/>
      <xsd:element name="item_count" type="xsd:int"/>
      <xsd:element name="shipTo" type="USAddress"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>

  <xsd:complexType name="USAddress">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
  </xsd:complexType>
</xsd:schema>
```

This schema defines the `PurchaseOrderType` object, which contains information about the order as well as a `shipTo` address made up of several fields. Here is the input data that corresponds to that schema:
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-01-21"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="po_2.xsd">
    <item>Widget</item>
    <item_count>15</item_count>
    <shipTo country="US">
        <name>John Doe</name>
        <street>123 North St</street>
        <city>Miami</city>
        <state>FL</state>
        <zip>33175</zip>
    </shipTo>
</purchaseOrder>

The output is a COBOL copybook with the following structure:

```
000010 01  PURCHASE_ORDER.
000020  02  ITEM  PIC X(30).
000030  02  ITEM_COUNT PIC 9(10).
000040  02  NAME   PIC X(30).
000050  02  STREET PIC X(30).
000060  02  CITY   PIC X(30).
000070  02  STATE  PIC X(30).
000080  02  ZIP    PIC S9(11).
```

The first step in creating this solution is to create the type trees. First, import the purchase_order.xsd file using the WebSphere Transformation Extender Schema Importer:

1. Open the WebSphere Transformation Extender Type Editor
2. Select **Import a type tree**
3. Select the **XML Schema** importer
4. Browse to **purchase_order.xsd** and then click **Next**.
5. The National Language default setting is correct, so click **Next**.
6. Type the name of your output file into the **File Name** field and click **Next**.
7. Click **Finish** when the type tree is created.
8. You may be prompted whether you'd like to open the type tree. If so, click **Yes**.
9. The created type tree should now be shown inside the Type Editor. You need to analyze the tree to make sure there are no errors before using it in a map. Select **Tree => Analyze => Structure Only**. You may observe warnings related to descriptions that were discarded, but these are normal.
10. Save the type tree and close it.

Next, perform similar steps to create the type tree for the copybook.

1. Select **Tree => Import**.
2. Select the **COBOL Copybook** importer
3. Browse to **po_copybook.cpy** and then click **Next**.
4. The Data Language settings are correct. Click **Next**.
5. Type the name of your output file into the **File Name** field and click **Next**.
6. Click **Finish** when the type tree is created.
7. You may be prompted whether you'd like to open the type tree. If so, click **Yes**.
8. The created type tree should now be shown inside the Type Editor. You need to analyze the tree to make sure there are no errors before using it in a map. Select Tree => Analyze => Structure Only.
9. Save the Type Tree and close it.

You have now defined inputs and outputs and are ready to complete the map.

1. Open the WebSphere Transformation Extender Map Designer and create a new Map Source file.
2. Select Map => New and call your map XMLtoCOBOL.
3. Right-click Input Cards and select New.
4. Set the CardName to input, select purchase_order.mtt as the TypeTree, select Doc as the Type, and choose po_xml_data.xml as the FilePath.
5. Right-click Output Cards and select New.
6. Set the CardName to output, select po_copybook.mtt as the TypeTree, select PURCHASE_ORDER as the Type and po_copybookdata.txt as the FilePath.
7. Expand the types in the From and To windows. This will show all the fields of the input and output cards. Drag the relevant input fields to the output fields.
8. Select Map => Build, and then Map => Run. To examine the run results, select Map => Run Results.
9. To run the map on DataPower, change the MapRuntime to WebSphere DataPower using the Map Settings dialog, then repeat Step 8.

**Tips for using WebSphere Transformation Extender Design Studio with DataPower**

The previous examples showed how to build type trees and maps that can be used on DataPower. Previous users of WebSphere Transformation Extender should note that certain functions are restricted on the DataPower runtime. To simplify the process for users, the Map Designer performs a check at build-time to make sure no restricted functions are contained in the map. A full list of these restrictions is provided in the WebSphere Transformation Extender Design Studio documentation.

**Conclusion**

This article showed how to get started creating type trees and maps using WebSphere Transformation Extender Design Studio and deploying them on WebSphere DataPower SOA Appliances. DataPower Appliances provide powerful functionality that is amplified when used with WebSphere Transformation Extender Design Studio.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code sample</td>
<td>XMLtoCOBOL.zip</td>
<td>27 KB</td>
</tr>
</tbody>
</table>
Related topics

- **Websphere DataPower SOA Appliances product page**
  Product descriptions, product news, training information, support information, and more.
- **Websphere DataPower SOA Appliances product library**
  Product announcements, case studies, white papers, and more.

© Copyright IBM Corporation 2007
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)