Testing XML Data with Informatica
Data Validation Option
Abstract

With Informatica’s Data Validation Option (DVO), you can reduce the complexity to test XML data and provide flexibility to test the data in both XML and non-XML data scenarios. This article describes how you can use DVO to test XML data.

Supported Versions

• Informatica Data Validation Option 9.5.2

Table of Contents

Abstract ............................................................................................................................................ 2
Supported Versions ......................................................................................................................... 2
Table of Contents ............................................................................................................................ 2
Introduction ...................................................................................................................................... 2
Functional Overview ........................................................................................................................ 2
Scenario 1 – Importing XML Definitions and Viewing in DVO ......................................................... 3
Scenario 2 – Using XML Sources Directly in Table Pair/Single Table Tests .................................. 5
Scenario 3 – Flattening XML Using Join View ................................................................................ 6
Scenario 4 – Editing Table and Column Names in Join View ......................................................... 8
Scenario 5 – Overriding XML Source File Information .................................................................... 9
Troubleshooting ............................................................................................................................. 10
Maximum Row Size Limit .............................................................................................................. 11

Introduction

XML is a commonly used format for moving data within and across enterprises. It is easily transported, machine independent, and can handle complex data relationships. But XML’s flexibility comes at a price. The hierarchical nature of XML makes parsing and processing more complex than table or flat file data. And testing XML data used in ETL or other application scenarios is much more tedious, time consuming, and error prone than flat file or relational data. XML hierarchies need to be traversed and often flattened in order to process the underlying XML data. When it comes to testing applications that leverage XML – e.g. testing whether XML data was loaded correctly into a relational table – the process can be complex. The XML needs to be read, flattened, and loaded into tables (typically all manually) and then tested against the relational structures.

Flattening XML typically is done via code (e.g. Java or some other convenient language) written by developers, with tools like PowerCenter, or in databases using SQL with XML extensions. Regardless of the method, this adds complexity and risk into the testing. And this is on top of the issues that already exist in manual testing for relational data.

What is needed is the ability for testers to work as easily with XML as they do with relational data, to be able to flatten it as need on the fly (i.e. no staging), and then create and execute tests quickly and easily. In short, what is needed is a way to remove the complexity of testing XML data, and yet still provide the flexibility to test in both XML and non-XML data scenarios. Informatica’s Data Validation Option does precisely that. DVO 9.5.2 (released in October 2013) enables QA engineers, analysts and developers to test applications that process XML in the
same way that they test applications with relational data. And just as DVO can easily test heterogeneous data scenarios (flat files and relational data, mainframe, cloud, on premise etc.), the same rules apply to XML. XML data can be tested with non-XML data with ease.
The following document provides a detailed description of how DVO can be used to test XML data.
For additional information please refer to the DVO user manual, particularly the chapter entitled XML Data Sources. Also refer to the PowerCenter XML Guide for more general information on processing XML data.

Functional Overview

XML source/target definitions can be imported into DVO from PowerCenter, just like any other supported source/target. i.e. via the metadata refresh functionality.
These XML sources and targets can be used in Views, Table Pairs and Single Tables for testing. Given that XML is hierarchical in nature, there are some enhancements in DVO specifically added to enable support for XML. They allow the user to easily flatten the XML data to make it accessible for testing via Table Pairs/Single Tables. These specific features include:
- The ability to select a Single XML Group\(^1\) in the XML hierarchy and perform tests on the fields in that group.
- The ability to create self-joins of XML Groups in a Join View to flatten the XML.
- The ability to automate the creation of a Join View to flatten XML directly for an XML Source

Once the XML is flattened, either via Group selection or Join View, the XML can be used in DVO like any other data.
XML use in DVO is bound by the same rules as XML use in PowerCenter. i.e. given that all processing happens in PowerCenter, PowerCenter must be able to parse and process any XML definitions and data being used via DVO.
The general rules are below:
- XML schema size should be smaller than 400 elements and less than 100KB in size.
- XML data file size should be less than 10MB.
- Complexity profile is limited to three (3) hierarchy levels with support for Complex Types: Sequence, Any, and Choice.
- Supporting XML import wizard is limited to creating 400 XML groups.
- PowerCenter is sensitive to the provided XML structure – attributes order for example is very strict while W3C XSD rules allows it to come in any order and not as defined in the XSD.
The remainder of the document provides detailed usage scenarios to explain the functionality in more detail.

Scenario 1 – Importing XML Definitions and Viewing in DVO

XML definitions are imported into DVO in the same way as any other supported source/target is imported. It can be at the folder level -- via Refresh Folder (Sources and Targets) – or at the repository level via Refresh Repository.

\(^1\) NOTE: The term “XML group” refers to a related set of elements at the same hierarchy level in and XML definition. In PowerCenter lingo, the term “XML view” is used instead of the term “XML group”, but they mean the same thing. In DVO, given that the word “view” is already used in the context of Join View, SQL View, Lookup View, the word “group” was used to minimize confusion.
Once imported the XML metadata (definitions) can be viewed by clicking on a Repository source/target via the navigator.

The details of the source are shown in the bottom right Properties tab.

This definition – Employee – has 4 groups: X_EMPLOYEES, X_EMPLOYEE, X_EMAIL, and X_PHONE. For each group, the elements in the group along with their type, precision and scale are displayed.
Scenario 2 – Using XML Sources Directly in Table Pair/Single Table Tests

In a Table Pair or Single Table, when an XML definition is used, the XML Group drop-down is enabled, showing the groups for that XML definition.

If you are using an XML definition directly in a Table Pair or Single Table, select the XML Group from the drop down to identify which data you want to test.

Once the XML Group is selected, the elements in that group are available for testing just like any other data source.

This method restricts you to only the elements in the particular group you select. i.e. in the above example, if the XML file has the structure shown below, only those fields under X_EMPLOYEE (i.e. EMPID, LASTNAME, FIRSTNAME etc. will be accessible. To access fields under one of the other groups, select one of those groups instead.

NOTE: The columns starting with XPK_ (e.g. XPK_EMPLOYEES, XPK_EMAIL) and FK_ (e.g. FK_EMPLOYEES, FK_EMPLOYEE etc.) are columns automatically added when the XML definition is imported into PowerCenter. These are primary and foreign key columns used to associate related data across different groups. E.g. across X_EMPLOYEE and X_EMAIL.

There are some limitations that must be observed for XML Group definitions. The key ones are listed below, but a more detailed description can be found in the PowerCenter XML Guide.

- An XML definition can have up to 400 groups.
- A group in a source definition does not require a key.
• A group can have one primary key.
• A group can be related to several other groups, and a view can have multiple foreign keys.
• A column cannot be both a primary key and a foreign key.
• A foreign key always refers to a primary key in another group. You cannot use self-referencing keys.
• A generated foreign key column always refers to a generated primary key column.

Scenario 3 – Flattening XML Using Join View

In Scenario 2, a single XML group was used in the tests. In order to select and test columns from across groups, the XML must be flattened by joining across the XML groups.

In PowerCenter, this is normally done manually by developers when they create mappings sourcing XML.

In DVO, this can be done using the Join View to create joins across the desired XML groups in the XML structure. There are two ways to create these Join Views for XML data.

• Manually creating joins using the standard Join View dialog.
• Using the Create Join View dialog specifically available to flatten XML sources/targets.

You are welcome to try both, but the rest of the scenario explains how to do it using the Create Join View dialog.

Right-click on an XML definition and select Create Join View.

The XML Join View creation dialog appears.
Fill in the fields in the dialog and select the groups you want to join.
NOTE: The Join View Table Alias and Group names will be used by default to prefix the actual column names of the elements in the generated Join View. Thus an element called EMAIL in a Group called CONTACT with Join View Table Alias called EMPLOYEE will be EMPLOYEE_CONTACT_EMAIL. You can override this in the Join View Table Alias field.

Once you've input all the information, click **OK** to create the Join View.

The Join View can be seen in the Join View section of the Navigator.

Looking at the Join View itself, there are 4 “tables” joined – i.e. one for each group in the XML definition – and all fields are selected for output.

You can edit any aspect of the view to suit your needs, such as reducing the number of output fields, changing their order, adding joins with other data sources (e.g. if a join is needed between the XML and a relational table for testing) etc.
Scenario 4 – Editing Table and Column Names in Join View

Once you've created a Join View to flatten the XML, you can edit the column names that are exposed by the Join View.

**NOTE**: This is general Join View functionality and is not specifically designed to be used with XML, but given the default naming convention when using XML, it is a common scenario to rename columns to make testing simpler.

If we use the example from Scenario 3, the *Select Output Columns* dialog looks like this:
The Alias column lets you override the name of the fields. Use this column to create shorter, more “friendly” names if desired.

The Include as Output Field column lets you select which columns to expose to Table Pairs and Single Tables when the Join View is used.

**Scenario 5 – Overriding XML Source File Information**

For a given XML definition, there can be multiple XML files that match that definition and need to be tested. This is similar to flat files. There may be one flat file definition (e.g. source or target definition), but there may be multiple flat files with data that are used for testing.

When a Join View is used to flatten an XML structure, it becomes a general definition to process any XML file that conforms to that structure. Thus, the file location and the file name (even if specified in the original dialog used to create the Join View) can change. It should be possible to use the same Join View definition to process any file that conforms to that structure.

To enable that, the connection information (file name and path) can be over-ridden when the Join View is used in a Table Pair or Single Table. E.g. click the Conn A (or Conn B) button to show and edit the Connection details.
NOTE: Only Join Views that flatten a single XML structure and do not include any other tables/file can be over-ridden this way. If the Join View includes other objects or tables, the over-ride is not possible.

Troubleshooting

Unlike relational tables and data, which are relatively straight forward, XML structures can be quite complex, and span multiple files or locations. In short, there are more “moving parts” when dealing with XML, and each part has to be correct and consistent with the other parts.

When XML errors are encountered, there are several immediate areas to consider and verify.

- Does the XML data match the schema?
- Does the data match the PC metadata definition?
- For multi-file schemas, are the files and relative file path references matching?
- Are all the expected files present?

The error messages displayed in the DVO GUI is the first place to look.

For example, the following error message very clearly indicates the data does not match the schema, i.e. an element in the XML file is not defined in the XML definition.

In the following example, the message indicates the expected XML file is not present. This means the file could be missing, filename could be misspelled, or the directory/path was entered incorrectly etc.
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-254</td>
<td>Error Message: ERROR: Workflow [DU_WF_Joined_EMPLOYEE_EMPLOYEE_114]: Execution failed. Please check the workflow log for more information. Message Context: <img src="image" alt="WorkflowExecutionState.waitForCompleted" /></td>
</tr>
<tr>
<td>PIFR_28080</td>
<td>XML Reader: Fatal Error <a href="image">The primary document entity could not be opened.</a></td>
</tr>
<tr>
<td></td>
<td>id=C:[Informatica8.1.0]server[info_shared][ScFiles/employees.xml]</td>
</tr>
</tbody>
</table>

Use of Multi-file schemas could also cause these types of errors if the relative file paths are not correct.
These are the easier problems to solve because they follow well defined patterns and error messages are reasonably specific.

## Maximum Row Size Limit

Often, when importing schema definitions into PowerCenter, string fields are imported with “infinite” length. This is simply a convenience when the maximum size of the field is not known in advance.

“Infinite” length strings are given a physical length in PowerCenter of 898989 characters. It is a strange number, and I’m sure there is a good reason for that.

In most cases, having many fields this size does not cause errors, because the maximum row size that PowerCenter can process is 100MB. But some transformations have restrictions on their maximum row size. The Sorter can handle a maximum row size of 8MB, i.e. significantly smaller than the maximum row size of PowerCenter in general.

Sorters are used in DVO generated mappings to sort data before Joins. This is a best practice when using PowerCenter as Joiners process sorted inputs almost an order of magnitude more efficiently than Joiners with the same data, but unsorted.

If an XML definition has multiple “infinite” string fields, the following error may appear in the session log when a DVO job fails.

>**SORT_40414 : Error: Total row size [12586408] in transformation [NAME_6] is more than the allowed maximum [8388607].**

This error is almost certainly caused by “infinite” length strings in the PowerCenter source/target definition.

To resolve the problem, update the PowerCenter XML definition causing the error, and replace “infinite” field lengths -- i.e. 898989 -- with field lengths based on the size of the actual data.

## Authors

Saeed Khan
Principal Product Manager