

Moving XML Files within SAS[®] and Maintaining Segment Hierarchy

By Sarmad Pirzada, HDS (Hybrid Data Systems)

Abstract: With XML becoming a standard for neutral data exchange medium, SAS[®] V9 offers the XML libname engine and SAS XML Mapper for moving XML data into SAS. Although the Mapper allows an efficient way of developing a map file for importing the multiple hierarchies in XML files, exporting a schema-dependent XML file is a challenge. We have used PROC TEMPLATE to export XML files back in the same schema it was imported in. This has allowed us to move data back to XML multiple-level hierarchy after SAS transformations. We also explored other options available to a SAS programmer.

Introduction: The SAS XML libname engine is production in SAS V9.1 and works like any SAS Access product for XML files with less than three hierarchies. A map file is needed for XML files with three or more hierarchies. Once the data is in a SAS dataset, it loses its hierarchical relationship and if the XML libname engine is used to export back to XML, the output file looks very different than the originally imported XML file. It is here that the XML libname engine differs from other SAS/Access products. In this paper I explore the various SAS and non-SAS options to write data back to XML. SAS provides two possible ways to do this: the SAS procedure, PROC CDISC (for biotech) and writing your own code for PROC TEMPLATE.

What is an XML File? Extensible Markup Language or XML is a new technology for web applications and data transfer established by the World Wide Web Consortium (<http://www.w3.org>) that has a standard for creating XML tags for business data transfer. Essentially, a tag for transferring zip code information could be written as : <zipcode>98199</zipcode>. A DTD or Data Type Definition is used to identify the tags used in an XML message. Within an XML file, the hierarchies have a parent and child relationship.

http://www.spec2000.com/Forum/2002/Presentations/6SPEC2000andXML_files/fullscreen.htm - Microsoft Internet Explorer

DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT S1CPOXMT (SPEC_HDR?,
S1CPOXMTcmd)>
<ELEMENT S1CPOXMTcmd (CCD, CIC, SPL,
ICR, POC, BNO?, QTL?, EVT?,
S1CPOXMT_ORDER+)>
<ELEMENT S1CPOXMT_ORDER (TNC, CPO,
PNR, QTO, UNT, UNP, SSD, SSM?, SHT?, QTN?,
SDC?, MFR?, PRI?, ACN?, DIS?, PKG?, LSE?,
PBR?, PDP?, VRN?, ACK?, POU?, CTN?, REM*)>
Include SPEC HDR
Include Common Support Data Dictionary
```

XML

```
<S1CPOXMTcmd>
  <CCD>S1BOOKED</CCD>
  <CIC>QF2</CIC>
  <SPL>81205</SPL>
  <ICR>USD</ICR>
  <POC>1</POC>
  <BNO>3</BNO>
  <S1CPOXMT_ORDER>
    <TNC>341</TNC>
    <CPO>EOIJ1234567</CPO>
    <PNR>65B2468</PNR>
    <QTO>1</QTO>
    <UNT>EA</UNT>
    <UNP>25.20</UNP>
    <SSD>20030715</SSD>
  </S1CPOXMT_ORDER>
</S1CPOXMTcmd>
</S1CPOXMT>
```

A DTD – Data Type Definition is used to identify the tags to be used in the XML message.

Sample XML File: An XML file is a flat file with tags. An XML file when opened in a notepad may look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Service.xsd"
Version="1.0">
  <PDR_Segment>
    <FHG>N</FHG>
    <RGC>AL</RGC>
    <HPR>UAL</HPR>
    <JDT>2001-07-01</JDT>
    <KSD>2001-07-31</KSD>
    <LON>Disarray Airlines</LON>
    <WHO>Disarray Airlines</WHO>
  </PDR_Segment>
  <ServiceDetails>
    <QID_Segment>
      <ZFR>S4956</ZFR>
      <XMC>797</XMC>
      <CIN>25398</CIN>
      <VEG>N550UA</VEG>
      <BFN>EMBRAER</BFN>
      <NIN>550</NIN>
    </QID_Segment>
```

A Hypothetical Typical Task for a SAS Programmer: Let's assume that your manager has asked you to work on a project to ftp XML files from a server, validate against a predefined schema, import the data into SAS PC, perform appropriate transformations, export rejected data back to XML and load data to MS/SQL Server that has an XML import engine. For such a typical scenario, the following tasks would be needed:

- **Validate the XML file against a Predefined Schema (XSD) file:** For this purpose I found limited software and applications, including the SAS XML Mapper. The SAS XML Mapper is a JAVA Appalet developed by SAS to map XML data that is not readable by assigning an XML libname engine. The Mapper does some high level validation of an XML file against a schema but has limited functionality as far as fixing corrupt files. Also, I am not sure how vigorously it matches the XML file against the XSD file. The software that I found most useful was **XML Spy**. This is a fairly expensive tool (\$900/seat) but has a number of useful features.
- **Develop an XML Map File:** The second task would be to translate the hierarchies and element attributes in an XML file into a map file. This easy to use interface allows users to drag and drop tables and columns from the left hierarchy window to the right panel, where each segment within a XML file is translated into a SAS table. The interface also allows some limited format changes. The XML Map, just like the XML Markup, is case sensitive. A tag must be uppercase and attributes lowercase. The XPATH is a formal description of each element like UNIX. Basically, the XPATH defines the hierarchy with the table and tells the libname engine where to find the element. Since within the XML file there can be one to many relationship, this information is very important if two elements have the same parent. The map file allows the assigning of keys (ordinals) that keep track of the one to many-relationship with the table hierarchy. A typical map file looks like this (note the upper and lowercase differentiation):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ##### -->
<!-- 2005-04-27T21:07:53 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 9.1.0300.20040709.2028 -->
<!-- ##### -->
```

```

<!-- ### Validation report                                     ### -->
<!-- ##### -->
<!-- Map validation completed successfully. -->
<!-- ##### -->
<SXLEMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="MaintAccount" version="1.2"
xsi:noNamespaceSchemaLocation="http://www.sas.com/xml/schema/sxle12.xsd">

  <!-- ##### -->
  <TABLE name="SCHEDULE3">
    <TABLE-DESCRIPTION>SCHEDULE3</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/TABLE/SCHEDULE3</TABLE-PATH>

    <COLUMN name="AP_EV_ID_NO">
      <PATH syntax="XPath">/TABLE/SCHEDULE3/AP_EV_ID_NO</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>
  </TABLE>
</SXLEMAP>

```

SAS XML Mapper

The screenshot shows the SAS XML Mapper application window. On the left, a tree view displays the XML structure with columns like ROC, OPR, RDT, RSD, RON, WHO, and ShopFindingsDetails. A callout box points to the 'MPN (11)' column with the text 'Move XML Columns from this Screen'. On the right, the 'Properties' pane shows details for the 'HDR_Segment' column, including its path and end path. A callout box points to the 'RCS_Segment' column in the SXLEMAP list with the text 'To this Screen'. At the bottom, a table view displays data for the 'HDR_Segment' table, with columns CHG, ROC, OPR, RDT, RSD, RON, and WHO. The table contains four rows of data.

CHG	ROC	OPR	RDT	RSD	RON	WHO
N	58960	TWA	2003-07-01	2003-07-31	Honeywell	American Airlines
N	88960	DEA	2002-05-01	2002-05-31	Goodrich	Delta Airlines
N	98960	NWA	2003-06-01	2003-06-30	Honeywell	Northwest Airlines
N	08960	GOL	2003-08-01	2003-08-31	Honeywell	GOL Airlines

- **Creating an XML Library and Processing Data:** The SAS code for setting up a library is straight forward as it appears in the text box below.

```
*Create XML Libname ;
*Get contents of directory for schema file: c: dir > contents.txt;
  filename SXLELIB 'C:\HDS_Desktop\Sample XML Data\Service.xml';
  filename SXLEMAP 'C:\HDS_Desktop\Sample XML Data\Service.map';
  libname SXLELIB xml xmlmap=SXLEMAP access=READONLY;
  proc datasets library=Sxlelib;
run;
```

- **Load Data to SQL Server:** For the purposes of this paper, the only comment I would make is that most databases allow loading XML data directly. This is significant if data could be written back to XML, there would be no need for SAS/Access module for that database. ***Herein lies another important purpose of this paper: how do you organize your XML output from SAS data in an appropriate schema that matches the schema of the target database?***
- **Organize SAS Data into Appropriate XML Structure-Option 1:** There are a couple of ways one can do this. One is to write extensive PROC TEMPLATE code to organize the SAS data in predefined hierarchy. This coding is very different when compared to SAS. I tried to create a three level hierarchy, and developed the following code with the help of SAS Tech Support:. This code gives you a flavor of the coding technique but is not a complete code-set for an XML file.

<pre>Proc template; define tagset tagsets.findings ; notes "SAS XML Engine output event model(interface)"; indent = 3; map = '<>&""'; mapsub = '/&lt;/&gt;/&quot;/&apos;/'; define event XMLversion; put '<?xml version="1.0"'; putq ' encoding=' ENCODING; put ' ?>' CR; break; end; define event XMLcomment; put '<!-- ' CR; put ' ' TEXT CR; put ' -->' CR; break; end; define event initialize; set \$LIBRARYNAME 'TABLE'; set \$TABLENAME 'DATASET'; set \$COLTAG 'column'; set \$META 'FULL'; eval \$is_engine 1; eval \$is_procpri 0; eval \$is_OUTBOARD 1; eval \$is_header_done 0; eval \$is_second_header 0; end; define event doc; start; trigger initialize; trigger XMLversion; break; finish; break;</pre>	<pre>define event leaf; start; eval \$is_engine 0; /* NOT ENGINE */ break / if ^cmp("Print", name); eval \$is_procpri 1; /* PROC PRINT */ eval \$regex prxparse("/\.(.+)/"); eval \$match prxmatch(\$regex, value); set \$TABLENAME prxposn(\$regex, 1, value); break; finish; break; end; define event output; start; break / if \$is_procpri ; eval \$is_engine 0; /* NOT ENGINE */ set \$TABLENAME name / if name; /* TABLE VIEWER */ break; finish; break; end; define event table; start; unset \$col_names; unset \$col_types; unset \$col_width; eval \$index 1; eval \$index_max 0; set \$TABLENAME name / if name; /* LIBNAME ENGINE */ set \$META XMLMETADATA / if XMLMETADATA ; /* LIBNAME ENGINE */ set \$SCHEMA XMLSCHEMA / if XMLSCHEMA ; /* LIBNAME ENGINE */ break; finish; break; end; run;</pre>
---	---

<pre> end; define event doc_head; start: break; finish: break; end; define event doc_body; start: break; finish: break; end; define event proc; start: break / if frame_name ; /* set by ODS statement use */ eval \$is_OUTBOARD 0 ; /* default for non-engine */ do / if cmp(HTMLCLASS, "OUTBOARD"); /* only the engine sets this */ eval \$is_OUTBOARD 1 ; else ; eval \$is_OUTBOARD 0 ; done ; break; finish: break; end; </pre>	
--	--

Use JAVA for Structuring Output from SAS-Option 2: (http://www.javaworld.com/javaworld/jw-09-2000/jw-0908-validation_p.html). A JAVA servlet can be developed to read XML file output from SAS, using a predefined schema. The most common way to handle data validation in JAVA involves simply coding the validation directly into the servlet, class, or EJB that deals with the data. In this example, validation is performed as soon as a parameter is obtained from a servlet. The advantage of this approach is that no proprietary software needs to be used for validation of the XML schema. Thus, XML output from SAS using just the XML libname engine can output, two-level XML files easily (as shown on the above example SAS code). Such files can then be processed by a custom JAVA servlet to validate and output XML. In this example, no ODS code would have to be written if you have JAVA skills.

Conclusions:

As XML becomes more extensively utilized and becomes a norm for data transfer, this paper outlines the intricacies of such a data exchange and options for SAS users. Fortunately, SAS Institute has developed the PROC CDISC for those in the biotech industries. Unfortunately for the rest of us, it will be a challenge to output XML if it is any format other than XMLTYPE=options. Although there is a tool to read such XML files but no SAS tool exists that can easily translate SAS data to complex XML output structures. This paper explores some of those approaches.

References and further readings

1. SAS[®] XML Libname Engine 9.1.3, User's Guide, SAS Institute, Cary NC, USA
2. Production: www.sas.com/rnd/base/topics/sxle82/prod82/libnamexml.html
3. Tutorial: www.sas.com/rnd/base/topics/sxle82/prod82
4. XMLMap: www.sas.com/rnd/base/topics/sxle82/exp82/index.html
5. Resources: www.sas.com/rnd/base/index-xml-resources.html

Acknowledgements

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. Other brand and product names are registered trademarks or trademarks of their respective companies.

® indicates USA registration

Author:

Sarmad Pirzada

SAS consultant

HDS

1420 5th Ave #2200

Seattle WA 98101

USA

Sarmad@hds-healthdatasystems.com.

206-898-5855

Sarmad Pirzada is a SAS consultant with HDS where he carries out and manages various SAS projects for local clients including Boeing, Washington Mutual Bank, Expedia and Path. He also consults for clients in CA (Tenet Health and Intuit Inc) where he provides applications support to browser-based SAS applications.