Import and Output XML Files with SAS

Yi Zhao

Merck Sharp & Dohme Corp, Upper Gwynedd, Pennsylvania

Abstract

XML files are widely used in transporting data from different operating systems or data storages. Processing XML files efficiently becomes one of the necessary technical skills for SAS programmers. This paper presents four aspects in importing and exporting XML files with SAS. They include: (1). Using XML engine LIBNAME and data step to import or output data between XML and SAS datasets. (2). Using PROC COPY. (3). Developing XMLMap to facilitate the conversion; and (4). Importing XML file to Excel and converting it to SAS dataset. Sample program codes and tips are provided and discussed.

Introduction

SAS programmers are used to imputing or outputting SAS datasets from or to other formats such as Excel spreadsheet, tab or space delimited ASCII file, XPT transport file. In recent years, another file format has become more and more popular – that is XML file. In pharmaceutical industry, the Food and Drug Administration (FDA) mandates pharmaceutical companies to submit certain data and documentation in XML format. Investigators are providing patient data with XML format. Consequently, it is imperative for SAS programmers to have good knowledge working with XML file.

What Advantages Does XML Provide?

XML is loaded with a mission of "carrying" data in a way that multiple applications on different platforms can recognize. Transporting a SAS data set in XML is the process of putting the file in a format in order to move it across hosts. Using the XML engine with the DATA step or with PROC COPY provides the following advantages:

- XML data is stored as text. Unlike SAS files, XML documents can be read and updated by using a text editor.
- XML documents can be imported into applications other than SAS applications. For example, an XML document can be input to an Oracle application. It can be delivered to the Web. It can also be restored as a SAS data set for continued processing. If compatibility with other programs is important for your data, the XML engine is recommended.
- The XML engine supports SAS 8 and later features. Unlike the XPORT engine, the XML engine supports SAS 8 features such as long names.

How Does XML Engine Work?

The XML engine works much like other SAS engines, such as XPT. That is, you execute a LIBNAME statement in order to assign a libref and specify an engine. You then use that libref in the SAS session where a libref is valid.

However, instead of the libref being associated with the physical location of a SAS library, the libref for the XML engine is associated with a physical location of an XML document. When you use the libref that is associated with an XML document, SAS either translates the data in a SAS data set into XML markup or translates the XML markup into SAS format. There are multiple approaches to inputting or outputting XML files using XML engine depending on the source data or personal preference.

Using XML Engine LIBNAME and Data Step

The syntax of the XML libname is similar to that of a standard SAS libname.

Inputting XML to SAS Dataset:

The following XML document, subjects.xml, will be converted to a SAS dataset in the example below:

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
   <SUBJECTS>
      <FirstName> James </FirstName>
      <Age> 10 </Age>
   </SUBJECTS>

   <SUBJECTS>
      <FirstName> Tom </FirstName>
      <Age> 12 </Age>
   </SUBJECTS>

   <SUBJECTS>
      <FirstName> Joe </FirstName>
      <Age> 15 </Age>
   </SUBJECTS>
</TABLE>
```

Here is the SAS code to create a permanent SAS dataset from the xml file:

```
libname myxml xml 'U:\XML\subjects.xml';
libname dat 'U:\data\';

data dat.subjects;
  set myxml.subjects;
run;

proc print data = dat.subjects noobs;
run;
```

PROC PRINT of data set dat.subjects:

```
             The SAS System

        FirstName          Age

        James               10
        Tom                 12
         Joe                15
```

Outputting SAS Dataset to XML:

```
libname myxml xml 'U:\XML\new_subjects.xml';
libname dat 'U:\data\';

data myxml.new_subjects;
  set dat.subjects;
run;
```

Please note that 'U:\XML\subjects.xml' is the physical location of the XML document for export or import. It is necessary to include the complete pathname, the filename, and the file extension. Enclose the physical name in single or double quotation marks. The .xml file specification must be specified.

In the LIBNAME statement above, an option xmltype= could be specified to generate brand-specific XML. For example, to export an XML document for use by Oracle, we could use the following LIBNAME:

```
libname myxml xml 'U:\XML\new_subjects.xml' xmltype = oracle;
```

By specifying the Oracle format, the XML engine generates XML that is specific to Oracle standards. The default xmltype is GENERIC, which is optional.

Using XML Engine LIBNAME and Proc COPY

Alternatively, we could use the COPY procedure to convert data set to or from an XML document.

```
libname myxml xml 'U:\XML\subjects.xml';
libname dat 'U:\data\';

proc copy in=myxml out=dat;
run;
```

In the code above, the libref MYXML points to the location of an XML document. The XML engine specifies that the XML document is to be read in XML format. The libref DAT points to the location where the contents of the XML document will be copied to. The PROC COPY statement copies the contents of the library that is specified in the IN= option to the library that is specified in the OUT= option.

Using XMLMap to Input XML Document

To successfully input an XML document using XML engine, SAS requires the inputted XML document to be well-formed with a rectangular structure. More specifically, the XML document must meet the following requirements:

      1. The root enclosing element (top-level node) of an XML document is the document container. For SAS, it translates to a library.
      2. The nested elements occurring within the container (repeating element instances) begin with the second-level instance tag.
      3. The repeating element instances must represent a rectangular organization. For a SAS data set, they become a collection of rows with a constant set of columns.

Here's the XML document used in the previous session that illustrates the physical structure that SAS requires:

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>            -- root enclosing element
   <SUBJECTS>     -- repeating element instance...first row
      <FirstName> James </FirstName>
      <Age> 10 </Age>
   </SUBJECTS>

   <SUBJECTS>      -- repeating element instance...second row
      <FirstName> Tom </FirstName>
      <Age> 12 </Age>
   </SUBJECTS>

   <SUBJECTS>      -- repeating element instance...third row
```

```
      <FirstName> Joe </FirstName>
      <Age> 15 </Age>
   </SUBJECTS>
</TABLE>
```

However, due to different sources and approaches an XML file is generated, many XML documents don't have the above physical structure required by SAS. Please see the sample XML document below.

```
<?xml version="1.0" ?>
<VEHICLES>
  <FORD>
    <ROW>
      <Model>Mustang</Model>
      <Year>1965</Year>
    </ROW>
    <ROW>
      <Model>Explorer</Model>
      <Year>1982</Year>
    </ROW>
    <ROW>
      <Model>Taurus</Model>
      <Year>1998</Year>
    </ROW>
    <ROW>
      <Model>F150</Model>
      <Year>2000</Year>
    </ROW>
  </FORD>
</VEHICLES>
```

Looking at the above XML document, there are three sequences of element start tags and end tags: VEHICLES, FORD, and ROW. As a result, SAS will not input it successfully. SAS would fail to identify columns of data and generate concatenated data. Here is what would happen when SAS reads this XML document:

1. The XML engine reads the XML markup until it encounters the <FORD> start tag, because FORD is the second-level instance tag.
2. The XML engine scans subsequent elements for variables. As a value for each variable is encountered, it is read into the input buffer. For example, after reading the first ROW element, the input buffer contains the values Mustang and 1965.
3. The XML engine continues reading values into the input buffer until it encounters the </FORD> end tag, at which time it writes the completed input buffer to the SAS data set as an observation.
4. The end result is one observation, which is not what we want.

Here is PROC PRINT output showing the concatenated observation.

```
                      The SAS System


              Model                   Year

              Mustang Explorer Tau    1965
```

To get separate observations, one solution is to use XMLMap. We change the table location path so that the XML engine writes separate observations to the SAS data set. Here are the syntax of XMLMap file and the process that the engine would follow:

```
<?xml version="1.0" ?>
<SXLEMAP version="1.2">
  <TABLE name="FORD">
  <TABLE-PATH syntax="xpath"> /VEHICLES/FORD/ROW </TABLE-PATH>
```

```
    <COLUMN name="Model">
      <DATATYPE> string </DATATYPE>
      <LENGTH> 20 </LENGTH>
      <TYPE> character </TYPE>
      <PATH syntax="xpath"> /VEHICLES/FORD/ROW/Model </PATH>
    </COLUMN>

    <COLUMN name="Year">
      <DATATYPE> string </DATATYPE>
      <LENGTH> 4 </LENGTH>
      <TYPE> character </TYPE>
      <PATH syntax="xpath"> /VEHICLES/FORD/ROW/Year </PATH>
    </COLUMN>
  </TABLE>
</SXLEMAP>
```

1. The XML engine reads the XML markup until it encounters the <ROW> start tag, because ROW is the last element specified in the table location path.
2. The XML engine scans subsequent elements for variables based on the column location paths. As a value for each variable is encountered, it is read into the input buffer.
3. The XML engine continues reading values into the input buffer until it encounters the </ROW> end tag, at which time it writes the completed input buffer to the SAS data set as an observation. That is, one observation is written to the SAS data set that contains the values Mustang and 1965.
4. The process is repeated for each <ROW> start-tag and </ROW> end-tag sequence.
5. The result is four observations.

The following SAS statements input the XML document and specify the XMLMap. The PRINT procedure verifies the results.

```
filename path 'U:\XML\path.xml';
filename map 'U:\XML\path.map';
libname path xml xmlmap=map;

proc print data=path.ford noobs;
run;
```

PROC PRINT output showing desired FORD data set

```
                    The SAS System

          Model                   Year

          Mustang                 1965
          Explorer                1982
          Taurus                  1998
          F150                    2000
```

In SAS version 9.1.3 a product called XML™Mapper is available which allows users to generate an XML Map by drag-and-drop interface. A separate license is required. Please refer to Reference for more details.

Importing XML File to Excel and Converting it to SAS Dataset

If you have trouble inputting an XML file to SAS dataset by using LIBNAME or XMLMap, there is another shortcut to achieve this task – using Excel to open the XML and then using PROC IMPORT to convert the Excel spreadsheet to SAS dataset.

XML features are available only in Microsoft Office Professional Edition 2003 and Microsoft Office Excel 2003. Here are the steps to input an XML file to Excel:

1. Select Open from the File menu and choose the xml file to be open.
2. On the pop up window 'Open XML', select 'As an XML list' and click OK.
3. Click OK on the next pop up window to let Excel create a schema based on the XML source data.
4. Click OK. The spreadsheet is displayed. Verify the spreadsheet against the XML file to ensure the conversion is correct.
5. Use PROC IMPORT to import the Excel spreadsheet to SAS dataset.

```
PROC IMPORT OUT=subjects DATAFILE="U:\subjects.xls" DBMS=xls;
  GETNAMES=yes;
RUN;
```

Conclusion

XML is a simple and flexible text format markup language playing an increasing role in the exchange of a wide variety of data. It has become a data interchange format standard. SAS software provides multiple ways to input and output XML files. Using XML engine and LIBNAME, combined with data step or COPY procedure, or an intermediate Excel conversion followed by IMPORT procedure, we could achieve our goal of data conversion successfully for most well-structured, generic XML documents. For non-generic XML documents, an XMLMap could be created to facilitate the conversion.

References

Shi-Tao Yeh, *Using XML™Mapper to Create An XMLMap*, Proceedings of the Northeast SAS Users Group Conference 2005. www.nesug.org/proceedings/nesug05/cc/cc8.pdf

SAS Help and Documentation, SAS Institute Inc.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Yi Zhao
Merck Sharp & Dohme Corp
301 N. Summertown Pike
North Wales, PA 19454
Phone: 267-305-7672
E-mail: yi_zhao@merck.com