



Using XML with the SAS System

Ian Sedgwick

SAS Education UK & Ireland

ABSTRACT

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

This presentation shows how SAS can be used to read and write XML documents. Examples will make use of the SAS XML LIBNAME engine to read and write standard XML structures, creating an XMLMAP with the new SAS XML mapper tool to read more complex XML structures and ODS tagsets to create customised XML.

INTRODUCTION

A SAS user may be interested in looking at an XML document as another data source on which to perform analysis. In which case they would be looking at how easy it is to transform the naturally structured hierarchical text file into a form which looks more like a SAS data set.

Alternatively, they may be interested in exporting the values of a SAS data set or the results of a SAS analysis into an XML document so it could be transferred to another operating system or imported into another application, such as a publishing application.

There are a number of tools in SAS which interoperate with XML, however this paper concentrates on just the SAS XML libname engine and the ODS MARKUP destination.

Importing XML Documents

XML documents can be read using the SAS XML libname engine. However unless options are specified the XML has to be structured in a very specific way:

```
<?xml version="1.0" encoding="windows-1252" ?>
<STUDY>
  <VITALS>
    <PATIENT> 1813 </PATIENT>
    <SEX> F </SEX>
    <VISIT> 1 </VISIT>
    <SYSTOLIC> 106 </SYSTOLIC>
    <DIASTOLIC> 74 </DIASTOLIC>
  </VITALS>
  <VITALS>
    <PATIENT> 1813 </PATIENT>
    <SEX> F </SEX>
    <VISIT> 2 </VISIT>
    <SYSTOLIC> 109 </SYSTOLIC>
    <DIASTOLIC> 73 </DIASTOLIC>
  </VITALS>
</STUDY>
```

This is what happens when the XML document is imported:

1. The XML engine recognizes <STUDY> as the root-enclosing element.
2. The engine goes to the second-level instance tag, which is <VITALS>, translates it as the data set name, and begins scanning the elements that are nested (contained) between the <VITALS> start tag and the </VITALS> end tag, looking for variables.
3. Because the instance tags <PATIENT>, <SEX>, <VISIT>, <SYSTOLIC> and <DIASTOLIC> are contained within the <VITALS> start tag and the </VITALS> end tag, the XML engine interprets them as variables. The individual instance tag names become the data set variable names. The repeating element instances are translated into a collection of rows with a constant set of columns.



Using this structure SAS data sets can be visualized simply by issuing the following statement.

```
LIBNAME STUDY XML "study.xml";
```



VITALS				
PATIENT	SEX	VISIT	SYSTOLIC	DIASTOLIC
1813	F	1	106	74
1813	F	2	109	73

Multiple SAS data sets can also be obtained in the same way from one XML document. It is important to realize that like all access engines what you are seeing in SAS is a view of the underlying XML document and not a physical copy of a SAS table. This has implications for how dynamic the underlying document is and also the performance of reading from a raw data file. Also because the XML document is a text file the engine has to read the file twice (by default) to determine basic attributes, like is it character data, numeric data, dates, times etc.

Supported File Formats

By default, the XML engine uses a GENERIC file format to read an XML document, however it can read other standard formats by using this option:

XMLTYPE=GENERIC (simple well formed XML)
ORACLE (Oracle8iXML)
OIMDBM (Open Information Model from the MDC)
EXPORT (alias for most common)
MSACCESS (Microsoft Access 2002 database)

Non-Standard structures for input

If the XML document does not follow the standard structure or is not one of the standard formats given above then you will need to do one of two things:

1. Transform the original XML document to one that does conform to the correct structure using tools outside of SAS (eg. XSL and a parser)
2. Create an XMLMap within SAS.

XMLMAPs

An XMLMAP is an external file which is formatted using the Xpath methodology and contains instructions on how to read the non-standard XML.

The XMLMAP markup is case sensitive with tagnames is uppercase and attributes in lowercase. The tagnames have a specific function , such as identify when row boundaries are or what attributes the SAS data set should have.



```
<?xml version="1.0" encoding="windows-1250"?>
<SXLEMAP version="1.2" name="Study" description="Study
Description">
  <TABLE name="DEMOG">
    <TABLE-PATH syntax="XPath">/STUDY/DEMOG</TABLE-PATH>

    <COLUMN name="PATIENT">
      <DESCRIPTION>Patient ID</DESCRIPTION>
      <PATH syntax="XPath">/STUDY/DEMOG/PATIENT</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>4</LENGTH>
    </COLUMN>

    ... More Code ...

  </TABLE>
```

<SXLEMAP> is the root container tag

<TABLE> defines the tables that are being created

<TABLE-PATH> defines the root location of the table

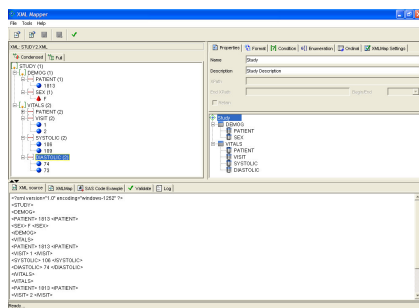
<COLUMN> defines a new column

<PATH> defines the root location of the column

<DATATYPE> specifies how to read the XML

<DESCRIPTION><TYPE><LENGTH> define the attributes storage in the SAS data set

XMLMAPs can be easily created using the new XML Mapper application.



The created XMLMAP file can be specified both as a library option or as a data set option.

Library Option:

```
libname study2 xml 'study2.xml' xmlmap='study2.map';
```

```
proc datasets lib=study2;
```

```
contents data=_all_;
```

```
run;
```

```
quit;
```

Data Set Option:

```
libname study2 xml 'study2.xml';
```

```
proc contents data=study2.demog(xmlmap='study2.map');
```

```
run;
```

So what are the benefits of the XMLMAP file ?

XML files can be read in one iteration as the specification to read them is already known. The path tags allow you to pick up attribute data as well as tag element data. You can choose to store numbers as a character field and handle dates and times correctly.



Exporting to XML from a SAS data set

Exporting can be done using the SAS XML libname. Code can be executed that transfers data from a SAS library to an XML library using the usual methods.

```
LIBNAME XMLLIB XML 'XML-Document';  
LIBNAME SASLIB 'SAS library location';  
  
PROC COPY IN=SASLIB OUT=XMLLIB;  
SELECT SAS data sets;  
RUN:  
  
DATA XML.table;  
SET SASLIB.table;  
RUN;
```

The same file formats that were supported for import are also supported for export with the addition of XMLTYPE=HTML . XML schemas can also be exported using the XMLSCHEMA='location of xsd' and XMLMETA can be used to indicate if you want to combine the schema and data in the same XML document.

Creating Customised Output

Customised output is created by using a specific Tagset. Tagsets are created using the TEMPLATE procedure and the resulting Tagset is stored in and Item Store.

Several Tagsets are supplied in the Tagsets directory of the SASHELP.TMPLMST item store:

1. SASXMISS - produces an empty element start tag and end tag for a missing value.
2. SASXMNMIS - does not generate tags for a missing value. That is, if a variable contains a missing value, the XML engine does not generate an element occurrence.
3. SASXMNSP - does not pad PCDATA with blanks.
4. SASXMOG - produces XML markup that is similar to the Oracle8iXML implementation used by Oracle but is more generic. This is the tagset used by the GENERIC format type.
5. SASXMOH - produces very simple HTML markup. This is the tagset used by the HTML format type.
6. SASXMOR - produces XML markup that is equivalent to the Oracle8iXML implementation, which is used by Oracle. This is the tagset used by the ORACLE format type.
7. SASFMT - produces XML markup for FORMAT and INFORMAT metadata generation.
8. SASXMDTD - for export only, produces an embedded DTD (Document Type Definition) in the body of the data markup. Note that DTDs are obsolete and XML Schema will be the only fully supported form in future releases.
9. SASXMXSD - for exporting only, produces an embedded XSD (W3C XML Schema) in the body of the data markup.
10. SASXMACC - MSAccess

Creating a Custom Tagset

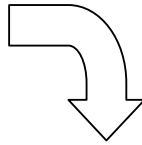
```
PROC TEMPLATE;

DEFINE TAGSET TAGSETS.STUDY
  /STORE=SASUSER.TEMPLAT;
  PARENT = TAGSETS.SASXMOG;
  NOTES "STUDY SPECIFIC OUTPUT";

DEFINE EVENT SASTABLE;
  START:
    PUTL "<STUDY>";
  FINISH:
    PUTL "</STUDY>";
END;
DEFINE EVENT SASROW;
  START:
    NDENT;
    PUT "<DATASET dsname=";
    PUTQ NAME;
    PUTL ">";
  FINISH:
    PUTL "</DATASET>";
    XDENT;
END;
DEFINE EVENT SASCOLUMN;
  START:
    NDENT;
    PUT "<VARIABLE ";
    PUT " varname=";
    PUTQ NAME;
  FINISH:
    PUTL "</VARIABLE>";
    XDENT;
END;

END;

RUN;
```



```
<?xml version="1.0" encoding="windows-1252" ?>
- <STUDY>
- <DATASET dsname="VITALS">
  <VARIABLE varname="DIASTOLIC">74</VARIABLE>
  <VARIABLE varname="SYSTOLIC">106</VARIABLE>
  <VARIABLE varname="VISIT">1</VARIABLE>
  <VARIABLE varname="PATIENT">1813</VARIABLE>
</DATASET>
- <DATASET dsname="VITALS">
  <VARIABLE varname="DIASTOLIC">73</VARIABLE>
  <VARIABLE varname="SYSTOLIC">109</VARIABLE>
  <VARIABLE varname="VISIT">2</VARIABLE>
  <VARIABLE varname="PATIENT">1813</VARIABLE>
</DATASET>
</STUDY>
```

Using the new Tagset

```
LIBNAME STUDY2 XML
  'study2.xml';
LIBNAME OUTXML XML
  'outvitals.xml'
  TAGSET=TAGSETS.STU
  DY;

PROC COPY IN=STUDY2
  OUT=OUTXML;
  SELECT VITALS;
RUN;
```

You should use customized tagsets with caution. If you alter the tagset when exporting an XML document and then attempt to import the XML document generated by that altered tagset, the XML engine might not be able to translate the XML markup back to SAS proprietary format.



Exporting Procedure Output to XML

Exporting procedure output is done using an ODS destination. You may already be familiar with writing procedure output to RTF or HTML using this technique. MARKUP is a new destination which takes advantage of supplied tagsets (in the same location to those shown earlier) and has the advantage that the supplied tagsets can be modified or you could create your own from scratch.

Basic syntax:

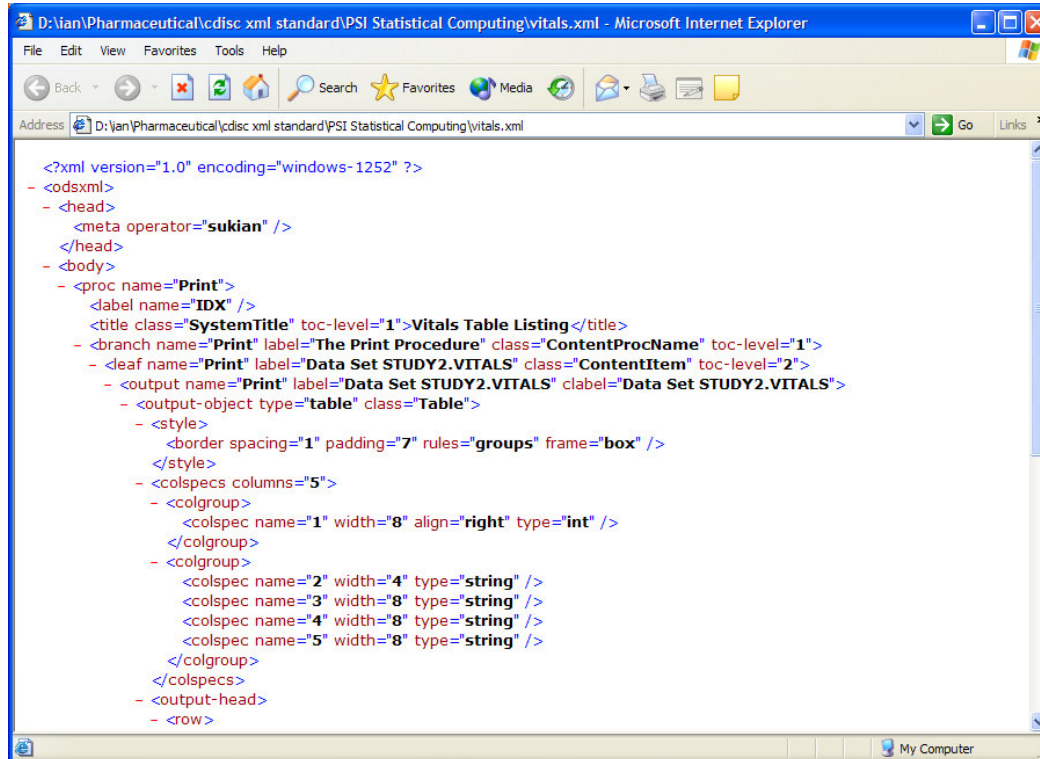
```
ODS MARKUP FILE='filename.ext'|FILEREFS TYPE=markup-type;  
... sas code ...  
ODS MARKUP CLOSE;
```

Without type= specified, the default output is the same as that produced by the ODS XML statement in 8.2.

Example

```
LIBNAME STUDY2 XML 'STUDY2.XML' XMLMAP='STUDY2.MAP';  
  
ODS MARKUP FILE='vitals.xml';  
TITLE "Vitals Table Listing";  
PROC PRINT DATA=STUDY2.VITALS;  
RUN;  
ODS MARKUP CLOSE;
```

Partial Output



```

<?xml version="1.0" encoding="windows-1252" ?>
- <odxml>
- <head>
  <meta operator="sukian" />
</head>
- <body>
  - <proc name="Print">
    <label name="IDX" />
    <title class="SystemTitle" toc-level="1">Vitals Table Listing</title>
    - <branch name="Print" label="The Print Procedure" class="ContentProcName" toc-level="1">
      - <leaf name="Print" label="Data Set STUDY2.VITALS" class="ContentItem" toc-level="2">
        - <output name="Print" label="Data Set STUDY2.VITALS" clabel="Data Set STUDY2.VITALS">
          - <output-object type="table" class="Table">
            - <style>
              <border spacing="1" padding="7" rules="groups" frame="box" />
            </style>
            - <colspecs columns="5">
              - <colgroup>
                <colspec name="1" width="8" align="right" type="int" />
              </colgroup>
              - <colgroup>
                <colspec name="2" width="4" type="string" />
                <colspec name="3" width="8" type="string" />
                <colspec name="4" width="8" type="string" />
                <colspec name="5" width="8" type="string" />
              </colgroup>
            </colspecs>
            - <output-head>
              - <row>

```

A DTD and XSL can be created using options on the ODS MARKUP statement.

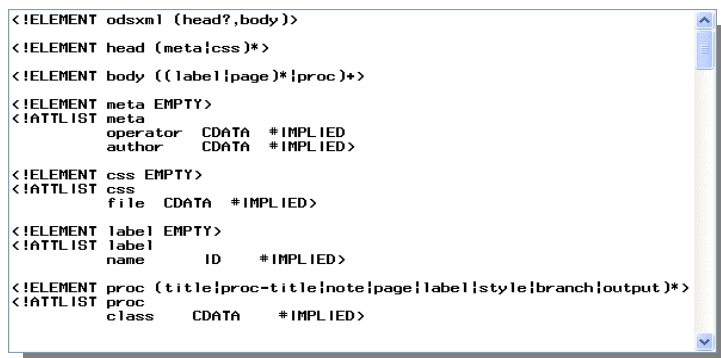
```

LIBNAME STUDY2 XML
'STUDY2.XML'
XMLMAP='STUDY2.MAP';

ODS MARKUP
  PATH=',' (url=none)
  FILE='vitals.xml'
  FRAME='vitals.dtd'
  CODE='vitals.xsl'
  STYLESHEET='mystyle.c
ss'
;
  TITLE "Vitals Table
Listing";
PROC PRINT

```

DTD



```

<!ELEMENT odxml (head?,body)>
<!ELEMENT head (meta|css)*>
<!ELEMENT body ((label|page)*|proc)*>
<!ELEMENT meta EMPTY>
<!ATTLIST meta
  meta      CDATA      #IMPLIED
  operator  CDATA      #IMPLIED
  author    CDATA      #IMPLIED
<!ELEMENT css EMPTY>
<!ATTLIST css
  css      CDATA      #IMPLIED
  file     CDATA      #IMPLIED
<!ELEMENT label EMPTY>
<!ATTLIST label
  label    ID          #IMPLIED
  name     ID          #IMPLIED
<!ELEMENT proc (title|proc-title|note|page|label|style|branch|output)*>
<!ATTLIST proc
  proc     CDATA      #IMPLIED

```



XSL

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/TR/REC-html40"
>

<xsl:output method="html"/>

<xsl:template match="odsxml">
  <HTML>
    <HEAD>
      <TITLE><xsl:value-of select="proc/title"/></TITLE>
      <LINK REL="STYLESHEET" HREF="mystyle.css">
    </HEAD>
    <BODY CLASS="Body">
      Table of Contents<HR/>
      <xsl:for-each select="body/proc">
        <xsl:apply-templates mode="toc"/>
      </xsl:for-each>
      <P><HR/></P>
      <xsl:apply-templates/>
    </BODY>
  </HTML>
</xsl:template>

<xsl:template mode="toc" match="*"></xsl:template>
```

ODS or Libname ?

ODS and the XML Libname Engine both create XML documents.

The disposition of the created XML document determines which method is used to create the XML file:

The XML Libname Engine may be better suited for data transfer.

ODS may be better suited for XML report and document production.

You can use the TEMPLATE procedure to create custom tag sets that can be used with either method.

CONCLUSIONS

- SAS delivers the power & flexibility needed to read, write and process all variations of XML
- SAS can read well structured XML using the SAS XML libname engine and more complex XML using the XMLMAP
- SAS can write XML using either SXLE or ODS
- Tag sets can be modified by creating and using a custom TAGSET
- SAS is committed to extending and enriching the XML functionality available within its software

REFERENCES



SAS 9.1 XML Libname Engine User's Guide

SAS 9.1 Output Delivery System User's Guide

Chapter 11 (TEMPLATE Procedure: Creating Markup Language Tagsets)

<http://support.sas.com/documentation/onlinedoc/sas9doc.html>

CONTACT INFORMATION

Ian Sedgwick
Principal Specialist Education Consultant
Wittington House
Henley Road
Medmenham
Marlow
Buckinghamshire
SL7 2EB
Telephone: +44 (0) 1628 404 224
Mobile: +44 (0) 7717 580 426
Facsimile: +44 (0) 1628 404 448
Email: Ian.Sedgwick@suk.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright ©2004, SAS Institute Inc. All rights reserved.