Paper TU09

The Developing Role of XML

David Shannon, Amadeus Software, UK

ABSTRACT

This paper discusses the use of extensible mark-up language (XML) within the SAS® 9. The impact to users of the SAS language is discussed and demonstrated.

The use of XML is positioned within the pharmaceutical industry by giving reference to regulatory guidance.

Briefly discussed will be the developments made by SAS Institute over of previous versions of their software with regards to the use of XML as a data source and output format.

Practical demonstrations of using XML as a data source and writing to output destinations will be given with SAS 9.

This paper is of relevance to all users of SAS who expect to be involved with the use of extensible mark-up language either as data source, transfer or delivery format.

INTRODUCTION

When working with a technology or language, I always feel more comfortable if I understand its origins and what it is trying to achieve, so let's start at the beginning. What is XML and where did it come from?

Like HTML, used in web pages, XML is a subset of Standard Generalised Mark-up Language (SGML), however unlike web pages XML is used to describe what information actually is, rather than its appearance.

XML was developed as recently as 1996 by a working group chaired by Jon Bosak (Sun Microsystems) falling under the umbrella of the World Wide Web consortium (www.w3c.org) – with the primary aims of allowing data to be published easily across the internet.

The goals of XML were *originally* to enable ease of use across the internet, to be widely supported by a variety of applications, easy to process, easy to create and be legible to the human eye; amongst other reasons.

As XML standards and rules are published by the open consortium W3C it enables anyone to implement the specification within their own systems and applications. This has enabled much greater integration between software and therefore the transmission, or sharing, of data than had easily been available previously. Along with consumer driven demand for open and integrated software, I believe this has been a key factor in the rapid success of XML.

Numerous software houses have integrated support for XML into their systems including Microsoft, Oracle and SAS Institute to name some key stakeholders in our work; indeed many software products are now orientated around XML, such as Microsoft® Office Word 2003 from which this paper is written!

It is the ability to swiftly move data between vendor independent databases and analytical tools, in addition to the use of the internet, which is our interest.

Reading data into the SAS System for our storage, analysis, reporting and even exporting of data now becomes the focus of this paper, however before this I should point out that XML is not necessarily the be-all and end-all of data formats.
XML does have some potential limitations. Its openness can be an issue when securing data is a priority. Also the size of XML files, relative to the amount of actual data they contain, is frequently found to be inefficient.

BETWEEN EDC & PDF

Specifically within the pharmaceutical industry, regulatory bodies such as FDA and organisations such as ICH suggest the use of open standards for the transmission of clinical submissions and their data.

Open standards means that anyone can request, or more often, download from the internet the specification or source code which makes up a format for a file, software etc. One such example is SAS Institutes XPORT file format, for which anyone can download C code the specification to create this format (http://support.sas.com/techsup/technote/ts140.html), even though SAS already supports it as a libname option.

The FDA has recommended that SAS data are transmitted to them in the XPORT file format. The XPORT, however, format does not support the use of long variable names and extended character variable attributes available after SAS 6.12. This is where the potential of XML comes to fruit.

One of the recently most frequently discussed pharmaceutical topics is that of electronic records and signatures. The guidance document 21 CFR part 11 has since 1997 discussed the transmission of electronic records and the use of XML.

Several guidance documents from both the FDA and ICH at least in part already discuss the use of XML, specifically 21 CFR part 11 and its associated guidance documents and not least the ICH multidisciplinary document “Electronic Common Technical Document Specification (ICH M2 EWG)”.

One of the fullest guidance documents currently making use of XML is that of “Electronic Transmission of Individual Case Safety Reports Message Specification”, of ICH M2 EWG.

THE ROLE OF XML

Hopefully by now you have a feel for what XML is used for and are yearning to see some examples of SAS at work.

Let’s start by examining some simple XML data with which we wish to work with in SAS. The sample shown below represents one row of data, immediately following the standard tag which identifies this document as XML.

```
<?xml version="1.0" encoding="windows-1252" ?>
- <TABLE> ①
  - <EFFICACY> ②
      <STUDYID>Amadeus Demonstration</STUDYID>
      <SITEID>1</SITEID>
      <USUBJID>HLR001</USUBJID>
      <SUBJID>1</SUBJID>
      <TRTCD>1</TRTCD>
      <INIT>HLR</INIT>
      <AGE>29</AGE>
③     <SEX>0</SEX>
      <BASELINE>54.1</BASELINE>
      <ENDPOINT>51.9</ENDPOINT>
      <ITT>1</ITT>
      <EE>1</EE>
      <FSTSCORE>1</FSTSCORE>
      <SNDSCORE>0</SNDSCORE>
    </EFFICACY>
```

Figure 1: Raw Data in XML

The “root node” in our XML data structure is represented here by the keyword “TABLE” (①). Nested within this is another node, or element, representing the name of our data (②), labelled “EFFICACY”. Nested within the efficacy node is an XML element③ is one element per variable (or column) in our source data.

Note that XML is a strict mark-up language and therefore each XML element ends with a close tag and the names of the elements are case sensitive.

The pattern highlighted (③) is repeated for each row of data before the outer XML elements are themselves closed at the end of the document

READING XML INTO DATA SETS

The SAS System allows plain XML data to be read in to SAS data sets with the XML libname engine from SAS V8.2 and newer.

The following code sample can be used to access a table of data stored in an XML document:

```
libname mydata xml "efficacy.xml";

proc contents data=mydata._all_;
run;
```

Figure 2: Reading Plain XML into SAS

The libname statement uses SAS XML libname engine option (SXLE) to point directly at the XML file. The contents procedure, reports the table found.

```
-----Directory-----

Libref:         MYDATA
Engine:         XML
Physical Name: C:\The Developing Role of XML\Data\Efficacy.xml

#  Name       Memtype
1  EFFICACY   DATA
```

Figure 3: Proc Contents of Data Generated from XML

Additionally, table and variable attributes are reported from Proc CONTENTS in the usual way except that the number of rows, creation and access times are unavailable. Furthermore SAS makes assumptions regarding informats and formats when reading these data. In SAS V8.2 it is not possible to associate existing formats with XML data.

```
data work.calculation;
set mydata.efficacy;
  delta = endpoint-baseline;
run;
```

Once the libref is available to our SAS session, tables of data within our XML document can be worked with in the normal way, via PROC and DATA steps. Consider the following simple code example:

Figure 4: Performing Calculations on XML Data

The above data step reads all 60 rows of data in the efficacy table directly from the XML file, calculates a new column and writes all the data out to a temporary data set in our SAS session.

The log file is shown below (run from Windows XP Pro, Pentium IV Xeon 3.0Ghz and 512MB RAM):

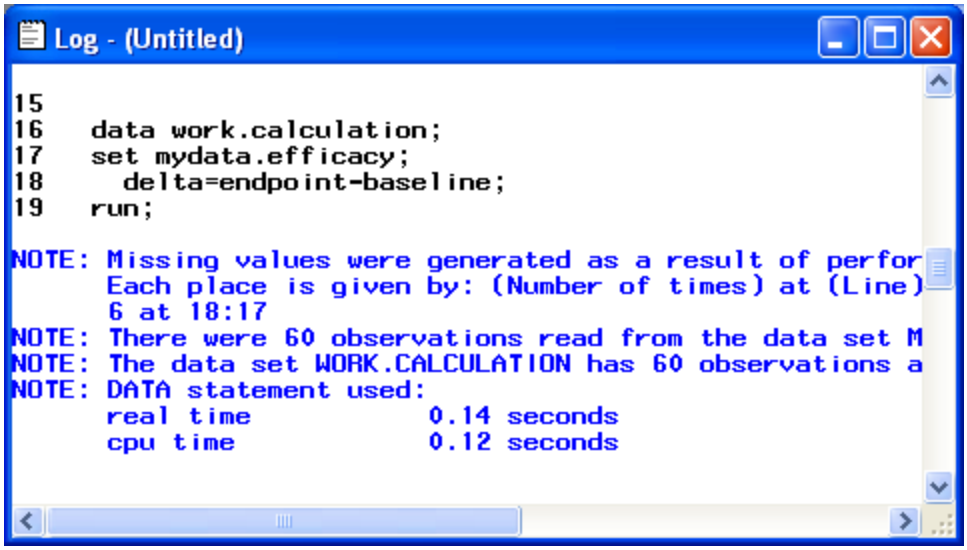


Figure 5: Performing Calculations on XML Data - Log File

Note the times taken to perform this step. Next the XML data is imported into a native SAS data set and the data step is again executed on the same PC, the log file is shown below:

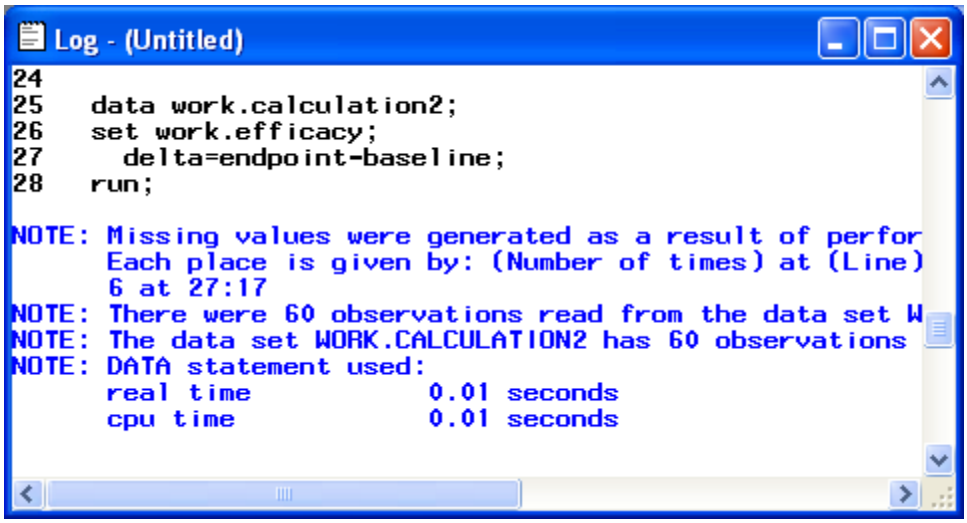


Figure 6: Performing Calculations on SAS Data - Log File

The speed taken to process these data are now virtually instantaneous. After some basic experimentation I am confident that, whilst this is strictly not a benchmark test, it is a consistent finding that reading data from directly from XML is considerably less efficient that reading from a native SAS format.

It is therefore clear that although we may be provided with data in XML format, it is most efficient to import those data into the native SAS format prior to performing any processing. PROC COPY can be used to perform this step.

WRITING SAS DATA TO XML

Data can be exported from SAS data sets to XML files as easily as importing data through the libname XML engine. The following simple code sample demonstrates this:

```
libname export xml "C:\BloodPressure.xml";

proc copy in=mydata out=export;
  select BloodPressure;
run;
```

Figure 7: Exporting SAS Data to XML

Again a libname statement is used with the XML engine pointing directly to the XML file we wish to create. Note that if the file exists its contents are overwritten.

INCREASED FEATURES THROUGH THE SAS VERSIONS

SAS users should be aware that much development has taken place during the lifetime of SXLE.

Its functionality within SAS V8.2 is restricted to importing what is referred to as "rectangular" XML i.e. all tags must appear within each repetition etc. Support for generating XML with the output delivery system, via the MARKUP destination, has experimental status within SAS V8.2.

Further changes and enhancements have been introduced through to the latest release of SAS at the time of writing this is SAS V9.1. The following features are available in 9.1 over 8.2:

- The ability to specify what type of schema the XML file confirms to can be specified with the XMLTYPE= option. Valid types in SAS 9 are:
    - GENERIC: The default value. Produces plain XML mark-up.
    - ORACLE: produces XML in the mark-up compatible with Oracle database.
    - EXPORT: Creates mark-up compatible with the formats most commonly used in industry. SAS Institute intends to upgrade this type as required in the future.
    - HTML: Creates HTML, i.e. web pages, from the data.
    - MSACCESS: Produces XML in the mark-up supported by Microsoft Access 2002 and newer.
- The XMLSCHEMA= option allows XML schema information to be stored in a separate file from the XML itself.
- It is usually required to export metadata, i.e. information about the data, along with the data itself. This is particularly important where formats are associated with data. The XMLMETA= option controls the export of metadata. Valid values are:
    - DATA: Specify this option to import or export only the data i.e. ignore any metadata.
    - SCHEMADATA: Imports or exports both the data and its metadata.
    - SCHEMA: Imports or exports only the metadata about the data.

  This option can be used along with the XMLSCHEMA option to control where metadata is written to.
- An option is provided to allow users to use a fileref in the libname statement rather than specify a string to the location. Use XMLFILEREF=*fileref* to do this.
- It is now possible to import several XML documents which are stored in a single file. Use the XMLCONCATENATE option to achieve this, although SAS Institute recommends using caution as this method is not standard XML.
- By default, all XML processors (not just the SAS System) should creating well formed or strict XML. It is possible to relax this rule with the XMLPROCESS= option. There are limitations with this option and I would not advise this without good reason!

- There are several options now available for controlling metadata repositories for XML maps. This is intended for environments where the SAS System is centrally managed by the SAS management console.

Some of these new features are essential for the level of flexibility frequently required in industry.

IMPORTING COMPLEX XML DATA USING SAS 9.1

As shown in the earlier example, the process of importing XML data into the SAS System is a simple case of using the SAS XML Libname Engine. This is fine in simple examples, such as those discussed so far, however what happens when we need to control the imported attributes of our data? This is when we must map our XML data to our data set.

```
filename mymap "efficacy.map";

libname readin xml "efficacy.xml"
        xmlmap = mymap ;

proc copy in=readin out=work;
run;
```

What is a map? A map is itself XML, stored in a file which contains the information about the data we wish to read in.

The map is stored in a text file, which is associated with our XML libname, via the XMLMAP= option, for example:

Figure 8: Importing XML with an XML Map

An example of such a map is shown below, which can be used when reading the efficacy data from the earlier example.

```
<TABLE name="EFFICACY">
  <TABLE-PATH syntax="XPath">/TABLE/EFFICACY</TABLE-PATH>

  <COLUMN name="STUDYID">
    <DESCRIPTION>Study Label</DESCRIPTION>
    <PATH  syntax="XPath">/TABLE/EFFICACY/STUDYID</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>21</LENGTH>
  </COLUMN>

  <COLUMN name="SITEID">
    <DESCRIPTION>Study Centre</DESCRIPTION>
    <PATH  syntax="XPath">/TABLE/EFFICACY/SITEID</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>integer</DATATYPE>
  </COLUMN>

  <COLUMN name="USUBJID">
    <DESCRIPTION>Unique Subject ID</DESCRIPTION>
    <PATH  syntax="XPath">/TABLE/EFFICACY/USUBJID</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>6</LENGTH>
  </COLUMN>
```

Figure 9: Contents of an XML Map

The above map shows just the definition of the file and first three variables to be stored in the data set, the screen show shown, displays the columns window (SAS V9.1) of the resulting SAS data set.

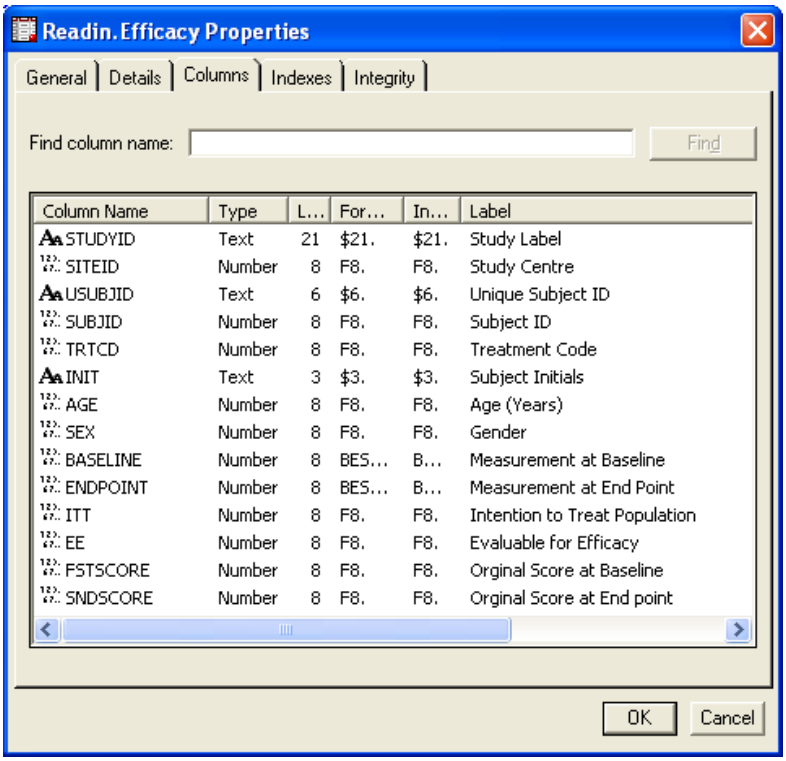


Figure 10: SAS 9 Variable Column Properties Window of Imported Data

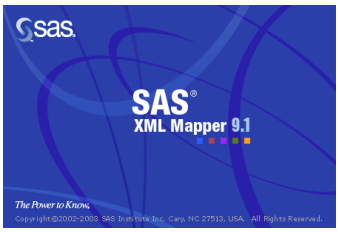
I suspect that are thinking this possibly going to more effort that the value of reading data from XML could be adding?

Well firstly a map only needs to be created once. It can then be used over and over, similarly to a well written SAS program.

Whilst the above map could be created by anyone using a simple editor such as Notepad, I hope most would agree that even after learning the appropriate syntax there is a high likelihood of typing mistakes not to mention a high level the time and effort required in this setup before the data will be accessible.

SAS XML MAPPER

Perhaps now I should come clean! Whilst possible, I don't really expect anyone to attempt creating a map file by hand. In fact I didn't create the efficacy map file by hand either.



Bundled with SAS 9.1 on volume 1 of the Client Side Components CD is an application called SAS XML Mapper (previously known as SAS XML Atlas in Beta releases).

This package is a freely available point-and-click tool which allows us to graphically create a map file before saving it to disk and using it in our SAS programs.

To create an XML Map the following minimal steps are taken:

- Launch the XML Mapper software by clicking the icon found under the Start - Program Files - The SAS System folders.
- Open the XML document you wish to create a map for.
- Change the default property 'Name' from SXLEMAP to a meaningful name for your data.
- Drag the table name from the condensed tab view onto the XML Map tree.
- Add columns by dragging from the condensed tab view onto the XML map tree.
- Use the Validate feature to ensure the validity of the XML Map before saving it for use within a SAS session.

Additional properties for both the variables and data can be configured within the XML Mapper. Control over the name and label (properties tab), data type, formats and informats (format tab) and valid values (enumeration tabs respectively) are available amongst others.

A screen shot of the XML Mapper at work when creating the Efficacy map file is shown below.

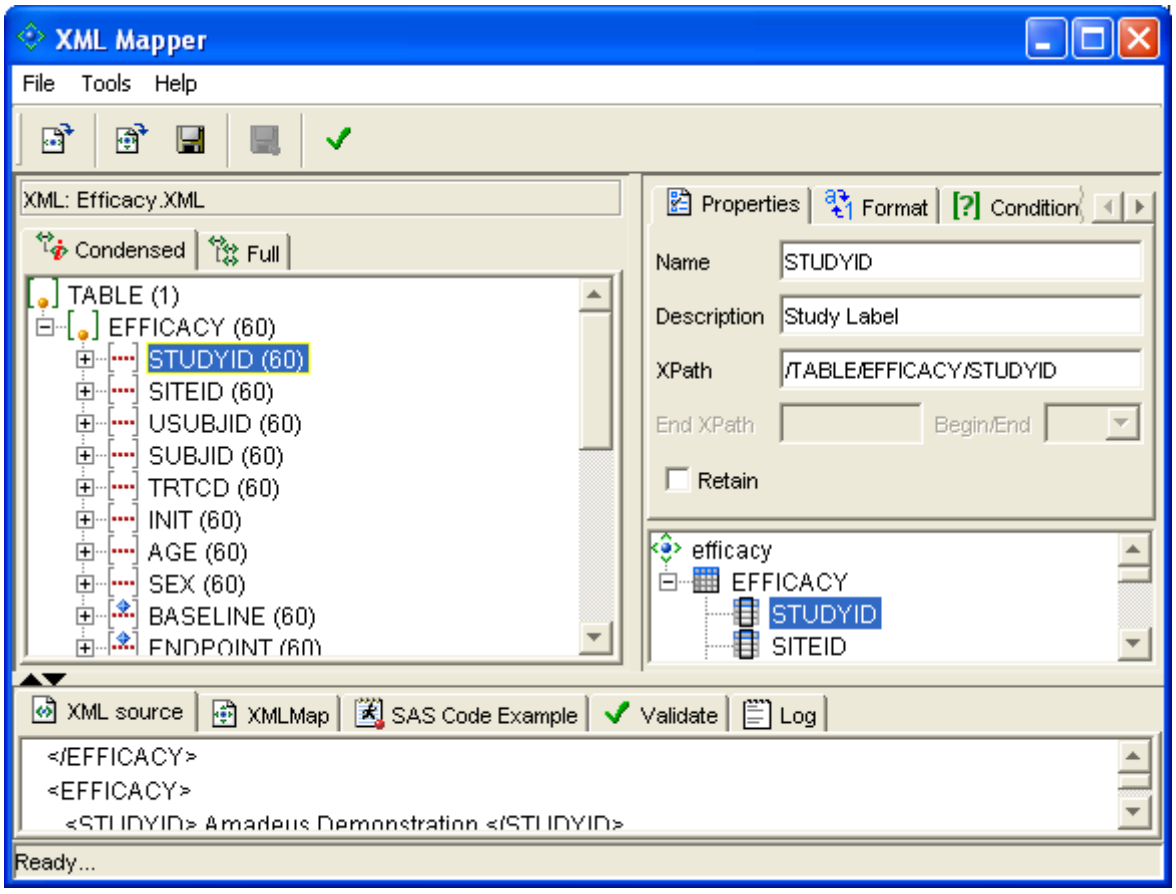


Figure 11: The SAS XML Mapper

MORE ON EXPORTING SAS DATA TO XML

So far I have tried to keep the use of XML to its simplest and whilst the examples discussed will work perfectly well, in reality it is more likely that XML will be used for data exchange between two systems, used over the internet or used between two parties – such as drug companies and regulatory bodies. In such cases it will be essential to include additional information other than simply the data, i.e. be able to describe what type of data is actually held in each XML element.

SCHEMAS

There are many acronyms associated with XML. In fact when scratching the surface of the XML world, like this paper, it probably appears a little bamboozling.

It is common place that XML schemas are defined for consistent structure to XML documents. This allows like industries, organisations or software packages to readily import the data held within XML documents, but also the metadata associated with the data in our XML documents. This information is referred to as a schema. The schema can be held either within the XML document or as a separate XSD file.

Easy examples are the schemas needed to import XML data into Oracle, or a Microsoft Access, each of which SAS V9.1 readily creates.

The FDA has a schema for individual case safety reports and plans for the future include defining schemas for all clinical data and reporting. The CDISC organisation is involved in this process and has already published its final Operational Data Model, schema (XSD) (and also DTD for backward compatibility).

WRITING SAS DATA TO XML WITH SCHEMA INFORMATION

Creating XML with schema information requires only the addition of some SXLE options.

The following is an example of creating XML from a SAS data set that is suitable for Oracle and Microsoft Access 2002, respectively:

```
* Export to Oracle;
libname oraxml xml "c:\eff2ora.xml"
               xmltype=oracle
               xmlmeta=schemadata;

proc copy in=rawdata out=oraxml;
 select efficacy;
run;

* Export to MSAccess;
libname msaxml xml "c:\eff2msa.xml"
               xmltype=msaccess
               xmlmeta=schemadata;

proc copy in=rawdata out=msaxml;
 select efficacy;
run;
```

When exporting data, SAS is using a one of a selection of tagsets held within the tagsets folder of the SAS templates. Furthermore, SAS programmers can create their own custom tagsets by using the TEMPLATE procedure in Base SAS; hence as new industry standard schemas are published they may be readily implemented in SAS.

To apply a custom tagset, use the TAGSET= option on the libname statement.

Figure 12: Exporting to XML with Schemas

DIRECTING PROCEDURE OUTPUT TO XML

This paper has demonstrated how the SXLE can be utilised to read and write data to and from XML directly to a SAS data set. It is also possible to make use of the output delivery system (ODS) through the markup destination to create XML from procedure output.

In my experience of the SAS System in the pharmaceutical industry there are perhaps four key ways in which reports are generated:

- Proc Report
- Proc Tabulate
- Proc Print
- Data steps with put statements

Each of these methods can be used to generate XML directly from the ODS MARKUP destination, or its alias ODS XML. The following example shows how the PRINT procedure creates XML:

```
* Proc Print;
ods xml body="c:\procprint.xml"
        frame="c:\procprint.dtd"
tagset=default;

  title "Simple Listing with Proc Print";
  proc print data=tdrx.demography;
  run;

ods xml close;
```

Figure 13: Directing Proc Print Output to XML

```
<?xml version="1.0" encoding="windows-1252" ?>
- <odsxml>
  + <head>
  - <body>
    - <proc name="Print">
        <label name="IDX" />
        <title class="SystemTitle" toc-
          level="1">Simple Listing with Proc
          Print</title>
      + <branch name="Print" label="The
          Print Procedure"
          class="ContentProcName" toc-
          level="1">
      </proc>
    </body>
  </odsxml>
```

The above code (Figure 13) produces both the XML file from the data and its associated Document Type Definition (DTD).

A section of the collapsed ODS XML is shown left (Figure 14).

Figure 14: Proc Print Generated XML

The following demonstrates how Proc Report can be used, in almost its simplest form to generate XML:

```
* Proc Report;
ods xml body="c:\procreport.xml"
        frame="c:\procreport.dtd" tagset=default;

  title "More Cable Report with Proc Report";
  proc report data=tdrx.demography nofs missing;
    column trtcd siteid subjid age weight enrolled terminated;
  run;

ods xml close;
```

Figure 15: Directing Proc Report Output to XML

A sample of the resulting XML is shown below:

Figure 16: XML Generated by Proc Report

CONCLUSION

The support for XML by various software vendors has gathered significant pace over the last few years. Particularly relevant in this paper are vendors such as SAS Institute, Microsoft and Oracle each of whom provided tightly integrated support for XML.

Within the pharmaceutical industry specifically, there are several advantages over the traditionally recommended data transfer format of XPORT, such as support for long character fields and other enhanced attributes of SAS variables. Several sets of regulatory guidelines, including the ICH common technical specification, have and are adopting XML as the suggested method of data transfer.

In general there are many advantages to XML such as the openness of data transfer. There are also possible considerations of securing XML and the relative efficiency of processing the XML data compared to native SAS data sets.

SAS Software supports XML from V8.2 via its libname engine option. Many enhancements have been introduced through to, at the time of writing the current release, SAS V9.1.

The recommended approach is to use SAS 9 whenever possible to benefit from the additional functionality available in the SXLE and the SAS XML Mapper software.

Using XML to access data should be considered as importing data and performed as a one off step at the beginning of a process due to the relative inefficiencies of XML compared to native SAS formats. Whilst accessing data from sources such as Oracle can be performed without the need to license additional SAS products, it is not recommended that XML be used in favour of, for example SAS/Access for Oracle; due to the advantages gained in security and efficiency.

REFERENCES

Operational Data Model, CDISC, Final Version 1.2, January 2004.
http://www.cdisc.org/models/odm/v1.2/index.html

ICH Common Technical Document (ICH eCTD), Version 3.2, February 2004.
http://www.ich.org

ICH M2 EWG: Electronic Transmission of Individual Case Safety
Reports Message Specification, Final Version 2.3 Document Revision February 1, 2001.
http://www.ich.org

SAS Institute
http://support.sas.com/software/91/engxmlwhatsnew900.htm

CONTACT INFORMATION

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

David Shannon
Amadeus Software Ltd.
Orchard Farm
Witney Lane
Leafield
OX29 9PG
Work Phone: +44 (0)1993 878287
Email: david.shannon@amadeus.co.uk
Web: www.amadeus.co.uk