ABSTRACT
XML is becoming the largest method of transporting and storing data. This is because XML is flexible and open. However, inorder to work with XML you need to understand XML and how SAS® uses and manipulates XML. In this paper you will learn XML structures, formats and best practices around manipulating XML data with SAS® XMLMapper and the SAS® XML Libname engine.

INTRODUCTION
XML, eXtensible Markup Language, is arguably the largest data transport mechanism today and is growing to be one of the largest storage mediums as well. This is because XML follows a standardized format, is easily readable by humans and machines and is easily extended without impacting communications.

• XML enables smart code. Because XML documents can be structured to identify every important piece of information (as well as the relationships between the pieces), it's possible to write code that can process those XML documents without human intervention.
• XML enables smart searches. Although search engines have improved steadily over the years, it's still quite common to get erroneous results from a search. If you're searching HTML pages for someone named "Chip," you might also find pages on chocolate chips, computer chips, wood chips, and lots of other useless matches. Searching XML documents for <first-name> elements that contained the text Chip would give you a much better set of results.
• XML is extensible. An applications can process XML documents even when extensions to the documents have been added that the application has not been made aware. A postal code for example, then the addition of <zip4> is added. The application will still process the <zipcode> element and ignore the <zip4> element until the application has been modified to handle the new addition. Even though the application has not had a chance to handle the new zip4 element, the application will still work with out error.
• XML follows schemas. A defined schema allows the document to be consistent so any application knows what to expect when it receives the information.

Because XML is so pervasive understanding how SAS® works with XML will help you in handling, manipulating and working with this common data source.

UNDERSTANDING XML
There are three common terms used in describing the parts of an XML document: tags, elements, and attributes. The, frontiersman, example XML snippet illustrates the usage of these terms:

A tag is the text between the left angle bracket ( <) and the right angle bracket ( >). There are the starting tags (such as <name>) and ending tags (such as </name>).

An element starts with the starting tag, includes the ending tag, and everything in between. In the sample above, the <frontiersman> element contains three child elements: <name>, <address>, and <city>.

An attribute is a name-value pair inside the starting tag of an element. In this example, state is an attribute of the <city> element. Each attribute must be enclosed in quotes as in the example <city state="TX">
• Every starting tag must have an ending tag. The ending tag tells the parser where to stop, processing that element.

• All the XML must be contained in one Element. This base element is known as the Root Element. The Root Element tells the parser where to stop processing the XML.

• Elements must not overlap. Overlapping Elements break the Hierarchy model of XML and make it impossible for a parser to understand what data goes where.

• Attributes must be in quotes. This allows the parser to know what is part of the tag and what is not. If the value of the Attribute contains a quote or double quote you can use the &quote entity which represents a double quote in XML.

• Elements are case sensitive <highway> and <Highway> are two different elements.

These basic rules need to be followed in order to have a valid XML, they also need to be understood by the end user so there is no confusion or to help troubleshoot problems with XML documents.

XML ARTIFACTS
There are many XML artifacts from namespaces to xref and xlink. In this paper we will cover a few common artifacts and the problems associated with the them.

Comments: in XML a comment is started by <!—and ends with -->. Comments can have anything in the comment except --. The only time a double dash is allowed is at the end of the comment.

Entities: The example below defines an entity for the document. Anywhere the XML processor finds the string &SGF; it replaces the entity with the string SAS® Global Forum.

<!-- Here's an entity: -->
<!ENTITY SGF "SAS® Global Forum">

The XML spec also defines five entities you can use in place of various special characters

• &lt; for the less-than sign
• &gt; for the greater-than sign
• &quot; for a double-quote
• &apos; for a single quote (or apostrophe)
• &amp; for an ampersand.

With XML, you can assign some meaning to the tags in the document. More importantly, it's easy for a machine to process the information as well. You can extract the postal code from this document by simply locating the content surrounded by the <postal-code> and </postal-code> tags, technically known as the <postal-code> element.

One important point about XML documents: The XML specification requires a parser to reject any XML document that doesn't follow the basic rules. Most HTML parsers will accept sloppy markup, making a guess as to what the writer of the document intended. Why is XML more restrictive? Because of its wider data integration goals and objectives. HTML is a sub-set of XML and if every XML parser was limited to only one type of XML this wouldn't be a problem but then again XML wouldn't be as effective either. So, to avoid the loosely structured mess found in the average HTML document, XML enforce document structure from the beginning. (a parser is a piece of code that attempts to read a document and interpret its contents.) Invalid, valid, and well-formed documents

There are three kinds of XML documents and one obvious unsupported document:

• Invalid Documents don't follow the syntax rules defined by the XML specification. If a developer has defined rules for what the document can contain in a DTD or schema, and the document doesn't follow those rules, that document is called Invalid.
• Valid Documents follow both the XML syntax rules and the rules defined in their DTD or schema.
• Well-formed documents follow the XML syntax rules but don't have a DTD or schema.
• Not Well-formed major error, they don't follow XML at all. The look like XML but are not. An example would be an XML document missing a Root Element.
UNDER THE COVERS OF SAS® PROCESSING XML

The XML libname engine is structured to handle all XML data within SAS. Under the covers the libname engine consists of 4 parts the Parser, Types, Monitor and then SAS. However, the XML engine works much like other SAS® engines. you execute a LIBNAME statement in order to assign a libref and specify an engine. You use that libref throughout the SAS® session where a libref is valid.

XML PARSER

An XML parser is an application that interprets the XML. There are 4 principle types of parsers and generally useful ones: the Document Object Model (DOM), the Simple API for XML (SAX), and their Java counterparts. There are pluses and minuses to each type of parser, SAS® uses the SAX parser to parse the XML. The SAS® libname engine is structured in such a way to optimize this performance and further enhance the processing of XML data in SAS.

NATIVE TYPES

The types contains the necessary information to parse the various supported native types such as ODM, WSDL and EXPORT. The native type is where the metadata about the XML and its schema are stored. This directs the parser to the xpath that needs to be parsed and also contains the necessary information to turn the hierarchical data into relational data.

MONITOR AND SAS

The Monitor acts does the error trapping and health check of the overall processing. It is the governor of the system and makes sure that the system is running properly. Then it is passed into SAS® where the data is treated like any other libname. A libref for the XML engine can be associated with either a specific XML document or, for the XML92 engine nickname, the physical location of a SAS® library in a directory-based environment. You can use a wildcard character in document names, such as *.xml. When you use the libref, SAS® either translates the data in a SAS® data set into XML markup, or translates the XML markup into SAS® format.

IMPORTING XML DATA INTO SAS

The SAS® XML libname engine processes the XML based on its knowledge of the XML. If the XML document is not of a type that is supported then XMLMapper needs to be used to map xml structures to their relational targets. The automap feature in XMLMapper can easily map the hierarchical data into relational form and will include Ordinal numbers that can be used to link relational data.

LIBNAME STATEMENT FOR IMPORTING DATA

In order to import XML data you simply need to create an libname statement and reference the appropriate XML type.
libname myxml xml 'C:\My Files\XML\frontiersman.xml';

This assumes the XML is a SAS® EXPORT or GENERIC form. If this was a non-standard form xml a map would be needed and the libname statement would look like this

filename map="C:\my files\xml\frontier.map";
libname myxml xml 'C:\My Files\XML\frontiersman.xml' xmlmap=map;

Once the libname is defined it is a simple as running any SAS® statement to get your results. Using the previous XML for frontiersman

proc print data=myxml.frontiersman;
run;

PROC PRINT Output of SAS® Data Set MYXML.frontiersman

The SAS® System

<table>
<thead>
<tr>
<th>Obs</th>
<th>STATE</th>
<th>CITY</th>
<th>ADDRESS</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TX</td>
<td>Alamo</td>
<td>1 Alamo Dr</td>
<td>Davie Crocket</td>
</tr>
</tbody>
</table>

EXPORTING DATA
You can also turn SAS® datasets into XML. The Libname engine will output SAS® datasets into a type supported by the XML libname engine including XMLMap. There are some limitations such as one table writing is supported as opposed to multiple table support. Below is an example:

To export the SAS® data set as an XML document that structures data hierarchically by division within each conference, an XMLMap is required. The only change to the existing XMLMap is to include the OUTPUT element. Notations in the XMLMap syntax are explained:

```xml
<?xml version="1.0" ?>
<SXLEMAP version="1.9"> 1
  <OUTPUT> 2
    <HEADING> 2
      <ATTRIBUTE name="description" 3
        value="Teams of the National Hockey League" />
    </HEADING>
    <TABLEREF name="TEAMS" /> 4
  </OUTPUT>

  <TABLE name="TEAMS">
    <TABLE-PATH>/NHL/CONFERENCE/DIVISION/TEAM</TABLE-PATH>

    <COLUMN name="NAME">
      <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@name</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>30</LENGTH>
    </COLUMN>

    <COLUMN name="ABBREV">
      <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@abbrev</PATH>
      <TYPE>character</TYPE>
```
To use an XMLMap to export the SAS® data set as an XML document, you must specify 1.9 as the XMLMap version number.

To use an XMLMap to export the SAS® data set as an XML document, you must include the OUTPUT element in the XMLMap. The OUTPUT element contains one or more HEADING elements and one TABLEREF element.

The ATTRIBUTE element, which defines additional file attribute information, specifies a name and description for the exported XML document.

The TABLEREF element, which references the name of the table to be exported, specifies the table TEAMS.

The following SAS® statements export the SAS® data set named NHL.TEAMS to an XML document named NHLOUT.XML, using an XMLMap named NHLEXPORT.MAP:

libname nhl 'C:\My Documents\myfiles';
filename out 'C:\My Documents\XML\NHLOUT.xml';
libname out xml92 xmltype=xmlmap xmlmap='C:\My Documents\XML\NHLexport.map';
data out.TEAMS;
   set nhl.teams;
run;

Here is the resulting XML document:

```xml
<?xml version="1.0" encoding="windows-1252" ?>
<!--    SAS® XML Libname Engine (SAS92XML)    
SAS® XMLMap Generated Output
Version 9.02.01B0D11292007
Created 2007-11-30T15:03:53
-->

<NHL description="Teams of the National Hockey League">
   <CONFERENCE>Eastern</CONFERENCE>
   <DIVISION>Southeast</DIVISION>
</NHL>
```
The ability to output XML then closes the loop as data can be imported manipulated then exported for use in other applications through-out the enterprise.

CALLING A WEB SERVICE
A really unique feature about the XML Libname engine is that is supports the Web Service Description Language, WSDL, format. This XML format allows one to call a Web Service that supports WSDL. Below is an example of usage of this format

```sas
filename WSDL "TestServiceAdd.wsdl" ;
libname  WSDL SAS92XML XMLtype=WSDL ;
proc datasets dd=WSDL details; run;
```

```
Member  Obs, Entries
#  Name         Type     or Indexes   Vars  Label
1  Add          DATA         0         2    WebService Add - Input parameters
2  AddResponse  DATA         0         1    WebService Add - Return parameters
```

```sas
proc contents data=WSDL.Add varnum; run;
```

NOTE: PROCEDURE DATASETS used (Total process time):
real time           20.84 seconds
cpu time            0.84 seconds

The CONTENTS Procedure

```
Data Set Name   WSDL.Add   Observations   0
Member Type     DATA       Variables    2
Engine          SAS92XML   Indexes      0
Created         Friday, January 01, 1960 12:00:00 AM
Last Modified   Friday, January 01, 1960 12:00:00 AM
Protection      Compressed  NO
Data Set Type   DATA       Sorted      NO
Label           WebService Add -
```
This type thus adds a whole new data source for integrating and manipulating data from inside and outside the company.

CONCLUSION
With the flexibility offered by XML it has become the predominate communication and integration messaging language. SAS’ XML libname engine combined with XMLMapper allows one to import virtually any type of XML
document into SAS® for integration and data manipulation purposes. SAS® can also export the data into XML format allowing for SAS® to be integrated with other systems within the Enterprise.

REFERENCES

RECOMMENDED READING

CONTACT INFORMATION (HEADER 1)
Contact the author at:
   Richard Foley Email: Richard.Foley@sas.com
   Tony Friebel Email: Tony.Friebel@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.