



CHAPTER

1

Getting Started with the XML Engine

<i>What Does the XML Engine Do?</i>	3
<i>Understanding How the XML Engine Works</i>	3
<i>Assigning a Libref to an XML Document</i>	3
<i>Importing an XML Document</i>	4
<i>Exporting an XML Document</i>	4
<i>SAS Processing Supported by the XML Engine</i>	5
<i>Frequently Asked Questions</i>	5
<i>Is the XML Engine a DOM or SAX Application?</i>	5
<i>Does the XML Engine Validate an XML Document?</i>	6
<i>What Is the Difference between Using the XML Engine and the ODS MARKUP Destination?</i>	6
<i>Why Do I Get Errors When Importing XML Documents Not Created with SAS?</i>	6
<i>Can I Use SAS Data Set Options with the XML Engine?</i>	6
<i>Why Does an Exported XML Document Include White Space?</i>	7

What Does the XML Engine Do?

The XML engine processes an *XML document*. The engine can

- export (write to an output file) an XML document from a SAS data set of type DATA by translating the SAS proprietary file format to XML markup. The output XML document can then be
 - used by a product that processes XML documents.
 - moved to another host for the XML engine to then process by translating the XML markup back to a SAS data set.
- import (read from an input file) an external XML document. The input XML document is translated to a SAS data set.

Understanding How the XML Engine Works

Assigning a Libref to an XML Document

The XML engine works much like other SAS engines. That is, you execute a LIBNAME statement in order to assign a libref and specify an engine. You then use that libref throughout 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.

Importing an XML Document

To import an XML document as a SAS data set, the following LIBNAME statement assigns a libref to a specific XML document and specifies the XML engine:

```
libname myxml xml 'C:\My Files\XML\Students.xml';
```

Executing the DATASETS procedure shows that SAS interprets the XML document as a SAS data set:

```
proc datasets library=myxml;
```

Output 1.1 PROC DATASETS Output for MYXML Library

Directory		
Libref	MYXML	
Engine	XML	
Physical Name	C:\My Files\XML\Students.xml	
XMLType	GENERIC	
XMLMap	NO XMLMAP IN EFFECT	
	Member	
#	Name	Type
1	STUDENTS	DATA

The PRINT procedure results in the following output:

```
proc print data=myxml.students;
run;
```

Output 1.2 PROC PRINT Output of SAS Data Set MYXML.STUDENTS

The SAS System					
Obs	STATE	CITY	ADDRESS	NAME	ID
1	Texas	Huntsville	1611 Glengreen	Brad Martin	755
2	Texas	Houston	11900 Glenda	Zac Harvell	1522

Exporting an XML Document

To export an XML document from a SAS data set, the LIBNAME statement for the XML engine assigns a libref to an XML document to be created from a SAS data set.

In the following code, the first LIBNAME statement assigns the libref MYFILES to the SAS library that contains the SAS data set Singers. The second LIBNAME statement assigns the libref MYXML to the physical location of the XML document that is to be exported from Myfiles.Singers:

```
libname myfiles 'C:\My Files\';

libname myxml xml 'C:\My Files\XML\Singers.xml';
```

Executing these statements creates the XML document named Singers.XML:

```
data myxml.Singers;
    set myfiles.Singers;
run;
```

Output 1.3 XML Document Singers.XML

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
  <SINGERS>
    <FirstName> Tom </FirstName>
    <Age> 62 </Age>
  </SINGERS>
  <SINGERS>
    <FirstName> Willie </FirstName>
    <Age> 70 </Age>
  </SINGERS>
  <SINGERS>
    <FirstName> Randy </FirstName>
    <Age> 43 </Age>
  </SINGERS>
</TABLE>
```

SAS Processing Supported by the XML Engine

The XML engine provides input (read) and output (create) processing. However, the XML engine does not support update processing.

The XML engine is a *sequential access* engine in that it processes data one record after the other, starting at the beginning of the file and continuing in sequence to the end of the file. The XML engine does not provide random (direct) access, which is required for some SAS applications and features. For example, you cannot use the SORT procedure or ORDER BY in the SQL procedure with the XML engine. If you request processing that requires random access, a message in the SAS log notifies you that the processing is not valid for sequential access. If this occurs, put the XML data into a temporary SAS data set before you continue. Note that the text of the SAS log messages will refer to invalid access attempts.

Frequently Asked Questions

Is the XML Engine a DOM or SAX Application?

Currently, the XML engine can be either a DOM application or a SAX application, depending on what you are doing:

- If the format type is either GENERIC (the default) or ORACLE, the XML engine uses a modified Document Object Model (DOM), which converts the document's contents into a node tree. However, for the XML engine, the node tree cannot be queried (traversed).

- If you are using an XMLMap to import an XML document, the XML engine uses a Simple API for XML (SAX) model. SAX does not provide a random access lookup to the document's contents; it scans the document sequentially and presents each item to the application only one time.

Note that for large XML documents for which you are simply using the format type `GENERIC` or `ORACLE`, if you are having resource problems, convert to using an XMLMap, which uses the SAX model.

Does the XML Engine Validate an XML Document?

The XML engine does not validate an input XML document. The engine assumes that the data passed to it is in valid, well-formed XML format. Because the engine does not use a DTD (Document Type Definition) or SCHEMA, there is nothing to validate against.

What Is the Difference between Using the XML Engine and the ODS MARKUP Destination?

Typically, you use the XML engine to transport data, while the ODS MARKUP destination is used to create XML from SAS output. The XML engine creates and reads XML documents; ODS MARKUP creates but does not read XML documents.

Why Do I Get Errors When Importing XML Documents Not Created with SAS?

The XML engine reads only files that conform to the format types supported in the `XMLTYPE=` engine option. Attempting to import free-form XML documents that do not conform to the specifications required by the supported format types will generate errors. To successfully import files that do not conform to the `XMLTYPE=` format types, you can create a separate XML document, called an XMLMap. The XMLMap syntax tells the XML engine how to interpret the XML markup into SAS data set(s), variables (columns), and observations (rows).

An exception is the HTML format type, which is supported only for export.

See Chapter 3, "Importing XML Documents," on page 29, Chapter 4, "Importing XML Documents Using an XMLMap," on page 45, "LIBNAME Statement Syntax" on page 83, and Chapter 8, "Creating an XMLMap," on page 95.

Can I Use SAS Data Set Options with the XML Engine?

Use SAS data set options with caution.

Note that while the `LABEL=` data set option no longer produces a warning message in the SAS log, the XML engine does not persist the information.

Why Does an Exported XML Document Include White Space?

The XML engine is in accordance with the Worldwide Web Consortium (W3C) specifications regarding handling white space, which basically states that it is often convenient to use white space (spaces, tabs, and blank lines) to set apart the markup for greater readability. An XML processor must always pass all characters in a document that are not markup through to the application. A validating XML processor must also inform the application which of these characters constitute white space appearing in element content.

When exporting an XML document, the XML engine adds a space (padding) to the front and end of each output XML element. Here is an example of an exported XML document that shows the white space.

Output 1.4 XML Document with White Space

```
<?xml version="1.0" encoding="windows-1252" ?>
- <TABLE>
-- <CLASS>
  <Name> Alfred </Name>
  <Sex> M </Sex>
  <Age> 14 </Age>
  <Height> 69 </Height>
  <Weight> 112.5 </Weight>
</CLASS>
```

The XML engine does not produce the special attribute `xml:space` for data elements but assumes default processing, which is to ignore leading and trailing white space.

You can remove the white space by specifying the SAS tagset `TAGSETS.SASXMNSP`. See “Using a SAS Tagset to Remove White Spaces in Output XML Markup” on page 73 for an example.

