Paper 173-28

XML? We do that!
Anthony Friebel, SAS Institute, Inc., Cary, NC

ABSTRACT
This paper is a tour of what's new in the development arenas for SAS® XML support. The features detailed include new options, new native support types, a demo of the SAS XML Mapper (a new XMLMap syntax generator GUI), new SAS ODS tagsets, and a preview of upcoming output generation enhancements to the SAS XML LIBNAME Engine.

INTRODUCTION
One of the most difficult things for me to deal with development-wise, aside from already being involved in XML's constantly changing usage and standards environment, is keeping track of the growing list of features and support items tending to blend together in a continuum that doesn't (necessarily) reflect the ship dates of a product. I cannot recall the last time that I spoke simultaneously of three different releases of SAS. The goal of this particular paper is to display the entire spectrum of Base SAS XML support, and to delineate the features available, as well as some caveats, across multiple product releases in the field.

Publication deadlines being what they are, it was nearing Christmas time when the finishing touches were put on this paper. Any coincidental references or organizational similarities to "A Christmas Carol" should be given the season's nod and summarily dismissed.

THE GHOSTS OF SXLE PAST (8.2)

SXLE
I think we've done a good job to date addressing the issues that arise in the field. The open mailing list[1] has provided a direct communication level with you in the trenches, and it has also proved to be a valuable resource for the next feature addition. There were some "all star" candidates in SAS 8.2 which I want to specifically mention and discuss.

TYPE RECOGNITION TIGHTENED
SXLE, in its native operating mode, is a two-pass engine. It makes one scan to determine the general construction details and detect data types; primary metadata considerations. The second scan extracts data content from the XML stream. There are a set of heuristics involved in that first pass used to determine what particular data element content "felt" like. Was it an integer? float? date? time? string?

Composite cases proved troublesome as delimiters became suspect. Phone numbers, SSN's, part numbers, etc. delimited by dots instead of dashes combined with a loose internal floating point check produced unexpected results. Dates not conforming to ISO-8601 standards were not recognized as dates. Those sorts of loopholes have been addressed and the native mode recognition is now much more robust. This should be welcome news for generic mode processing users.

ACCESS VIOLATIONS FROM UNSUPPORTED TYPES
SXLE was also very narrow-minded in the level of supported input types. Feeding it a non-supported type was usually met with a generic "DESCRIBE ERROR" at best, and at times, a general crash because things just weren't the way they were supposed to

be constructed. We've made a concerted effort to diagnose improper constructions and become more fault-tolerant in the upcoming releases. If an input stream doesn't seem to be constructed properly, you should now see a log message indicating such. And if not, I should see an e-Mail in the engine folder[1].

PARSER
Parsing of the XML stream is often taken pretty much for granted, but there are, inevitably, the rule bending, mind numbing, imagination stretching cases that crop up from real world scenarios. We've consolidated support of XML parsing "under the covers" and now use a new, improved version which other internal applications also rely upon. This will provide operation more consistent with other XML-based offerings and a sound base upon which to continue to build our functionality.

LONG LINES WITHOUT A CARRIAGE RETURN
Why would some applications generate a 2M (megabyte) stream of brackets and characters, but never emit a single carriage return in the entire wide swath? XML was (originally) intended to be human-readable which *tends* to mean white space and line breaks to cue the human through the document. However, human-readable has evolved to mean machine-readable, and those same visual elements are now simply a nicety, rather than a necessity. Likewise, we've made that internal processing adjustment.

ESCAPE SEQUENCES
Special characters are always problematic in computerized data processing. But, at times, it seems to be more a problem for the operators, than the actual machines. According to the XML specification, certain characters have specific escape sequences which are to be substituted in their place should they occur. Shortcuts like "#x27;" aren't expected when a single quote (') is to be replaced with the specification dictated string "'" The humans responsible saved one byte per occurrence in a stream how many megabytes long with that clever little trick. We have likewise become more clever, and have evolved the parsing mechanisms to accept the variant specifications.

CDATA SECTIONS
The ultimate special character handling avoidance mechanism is supported by the parser. *'nuff said.*

INTERNATIONAL CHARACTER SETS
The parser also now recognizes and is sensitive to national characters contained in an XML stream. One additional caution is required here. While we're much more aware of encoding issues than in prior incarnations, documents containing national characters should always begin with an XML declaration line which includes the encoding= attribute.

XMLMAP
A first quarter 2003 re-release of the XMLMap extension to SXLE, provides production status for the XMLMap input extension functionality.

EQUIVALENT V9 FUNCTIONALITY
A hybrid of the original and new, added functionality, the production extension provides a SAS 9.0 equivalent feature set to the 8.2 XML product.

As support levels have evolved, so too has the XMLMap syntax. We have decided on a version controlled parsing mechanism as the vehicle for new feature functionality introduction. Each version level dictates what feature set is available.

1.0	SAS 8.2 (and prior)
1.1	SAS 9
1.2	SAS 9.1

The original XMLMap version was dubbed "1.0" and represents primarily the SAS 8.2 capabilities. An updated release, the "1.1" represents 9.0 level capabilities. There are XML markup differences between the two versions and, as expected, one version syntax is mutually exclusive of other versions. Since it did not originally correspond, we chose not to use SAS software release numbers as syntax versions. We'll talk more about the new "1.1" features in the Future section to follow.

NEW SYNTAX
There are some fundamental changes in the "1.1" XMLMap markup that will be noticed immediately by users. The first, is a change to more generic terminology which is void of syntax dependency.

(X)PATH
The tags denoting paths (locations) in the XML stream are generic, and now indicate only PATH, rather than Xpath (the W3C standard upon which they are built). An optional **syntax=** attribute has been likewise added to indicate the particular specification to which the PATH conforms.

Old form :
```
<XPATH>
/NHL/CONFERENCE/DIVISION/TEAM@NAME</XPATH>
```

New form :
```
<PATH syntax="Xpath">
/NHL/CONFERENCE/DIVISION/TEAM@NAME</PATH>
```

or, simply
```
<PATH>/NHL/CONFERENCE/DIVISION/TEAM@NAME
</PATH>
```

"OLD-TIMER" UNDERSCORES
The second change is younger inclinations having prevailed and the underscores in tags have thus given way to the currently preferred XML markup trends using the dash character. OK. I admit it. I can still spell M-V-S. Please see Appendix A for a side-by-side comparison of syntax elements and release levels.

Old form :
```
<TABLE_XPATH>/NHL/CONFERENCE/DIVISION/TEAM
</TABLE_XPATH>
```

New form :
```
<TABLE-PATH>/NHL/CONFERENCE/DIVISION/TEAM
</TABLE-PATH>
```

THE GHOSTS OF SXLE PRESENT (9.0)

SXLE
Limited availability played down the changes that the product

offerings had undergone for SAS 9. The 8.2 bumps in the road had been smoothed and new functionality was added to address the growing field requirements of XML processing.

OIM XMLTYPE DEPRECATED
Mergers, acquisitions, abdications, are the way of the business world. The Metadata Coalition and its Open Information Model have also gone the way of bobby socks, hula hoops, and the Slinky, replaced by bigger, better, and improved offerings from new parent companies or consortiums. While it will continue (for a time) to be available as an XMLtype= operand, support for the OIM is being discontinued. Focus will shift to newer, more current offerings and users should begin to move away from the OIM XMLtype. There are also updates to other native XMLtypes coming (see below) which may provide equivalent or better functionality.

XMLSCHEMA OPTION RENAMED XMLMETA
The XMLSchema LIBNAME option was introduced when the W3C XML Schema was a draft proposal, but even then was probably a slight misnomer. SXLE users recognize the option as a way to control generation of additional metadata markup in supporting XMLtypes; format and informat information from OIM, header columns from HTML, etc. In an effort to separate the current functionality from upcoming support, the XMLSchema option has been renamed to a more corollary XMLMeta moniker. In SAS 9 programs, the old XMLSchema option is no longer accepted on the LIBNAME statement and will generate a warning in the SAS log. The functionality of the option remains the same.

NEW TAGSETS
The conversion to utilize Output Delivery System (ODS) tagsets for generated engine output was one of the fundamental architecture changes introduced to SXLE in SAS 8.2. Tagset customization permits user control of generated XML markup within a tabular construction constraint. It is possible to completely change the markup generated by SXLE via a custom ODS tagset. The following examples, taken from field experience, were common requests and are now available tagsets in SAS 9.

DTD AND SCHEMA GENERATION
It may be necessary to describe XML markup content generated by SXLE to other XML-enabled applications. A popular field request, these two tagsets produce an elementary embedded DTD[2] (Document Type Definition) or XSD (W3C XML Schema) in the body of the data markup. The elementary forms produced should be digestible by most XML-accepting applications. For DTD generation, on the LIBNAME statement, specify

```
LIBNAME myxml xml 'path-to-my-xml-data'
tagset=tagsets.sasxmdtd ;
```

For XSD generation, on the LIBNAME statement, specify

```
LIBNAME myxml xml 'path-to-my-xml-data'
tagset=tagsets.sasxmxsd ;
```

NO WHITESPACE IN PCDATA
If I had a dollar for every time someone has tried to wave a "non-conforming" finger at the engine for this particular markup behaviour, I'd be singing harmony with Steven Page and the rest of the BNL crew, and wearing a fur coat. But, not a real fur coat, that's cruel. Fact is, the problem most often lies with conflicting definitions of "default" behaviour among applications, and not specification conformance, and I'll make only footnote comment to the specification verbiage[3]. If you are thusly victimized by

application incompatibility, this tagset may prove of some benefit. On the LIBNAME statement, specify

```
LIBNAME myxml xml 'path-to-my-xml-data'
tagset=tagsets.sasxmnsp ;
```

MISSING TAG GENERATION (OR NOT)
SXLE changed the generated markup for a MISSING value between SAS 8.1 and 8.2 in favour of the shorthand notation **<tag />** which had come into vogue. However, some application parsers might still expect the old open-close element **<tag></tag>** form. You can return to yester-year (and the open-close markup), by specifying

```
LIBNAME myxml xml 'path-to-my-xml-data'
tagset=tagsets.sasxmiss ;
```

on the LIBNAME statement. To remove generation of tags for MISSING values entirely, specify

```
LIBNAME myxml xml 'path-to-my-xml-data'
tagset=tagsets.sasxmnmis ;
```

on the LIBNAME statement. The latter does not generate an element occurrence at all in the target markup if the column contains a MISSING value.

XMLMAP
Limited availability played down the changes that the product offerings had undergone for SAS 9. The 8.2 bumps in the road had been smoothed and new functionality was added to address the growing field requirements of XML processing.

CONDITIONAL SELECTION
The conditional selection syntax of the column PATH element has been expanded in the version 1.1 specification. You can now select pcdata or an attribute value based on attribute value. This enhancement rounds outs the total selection criteria; pcdata, conditional pcdata, attribute value, and conditional attribute value. All selections must be based on, and terminate with, the "current" element.

Accepted forms of PATH and conditional selections

```
<PATH>/LEVEL/ITEM</PATH>
<PATH>/LEVEL/ITEM[@attr2="value"]</PATH>
<PATH>/LEVEL/ITEM@attr</PATH>
<PATH>/LEVEL/ITEM@attr[@attr2="value"]</PATH>
```

NTH OCCURRENCE
The column PATH element expression also supports the position() function for accessing recurring element content of the same named element. Parallel to GENERIC mode generating suffixed element names for recurring elements, the expression enhancement allows creating multiple distinct columns for recurring elements in your XML input stream.

```
<PATH>/LEVEL/RECURS[position()=2]</PATH>
```

It should be noted that the recurring position() expression must based on, and terminate with, the "current" element. Position() is only supported for extraction of pcdata content.

ORDINAL GENERATION
Generation of sequential valued columns is now possible. Incrementing (and decrementing), as well as reset to zero

operations are available to a special column type we call an ordinal. These "computed" columns can be useful for discriminating values selected from recurring patterns in an XML input stream, creating distinct key values, or for any other situation where a counted value is desired. An ordinal is defined by the presence of an **ordinal=** attribute on the COLUMN element.

```
<COLUMN name="foo" ordinal="yes"> …
```

New INCREMENT-PATH, DECREMENT-PATH, and RESET-PATH elements are available in the COLUMN syntax group. INCREMENT and DECREMENT are mutually exclusive controls.

ATLAS (BETA)
Atlas is an XMLMap support tool introduced in SAS 9. It can be found on the client-side installation CD-ROM for SAS 9. Atlas is a Java-based, stand-alone application designed to remove tedium from the generation of XMLMap syntax, as well as provide a level of data exploration via its graphical user interface construction. The Atlas tool is considered a BETA in this release.
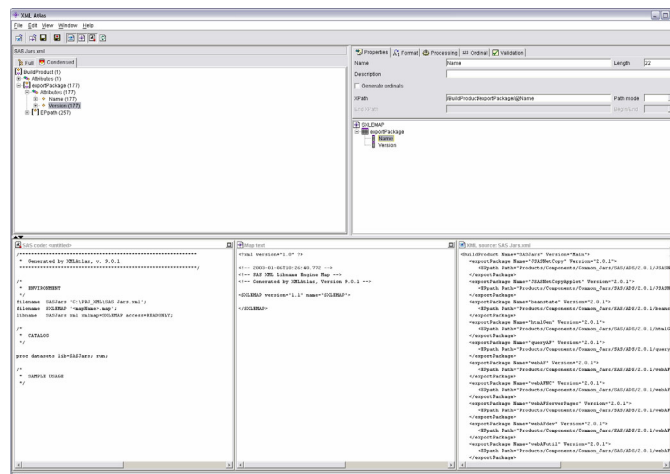
ASSIST TOOL FOR XMLMAP GENERATION
Atlas' primary task is the generation of XMLMap syntax. It displays the input XML sample in a text pane, as well as, in an Explorer-like graphical tree. Additionally, Atlas offers a "condensed" graphical tree pane, an XMLMap construction progress display pane, and a sample SAS program display pane.

The "condensed" pane is a tree display of data patterns occurring within the sample XML content. It tracks detected data type, maximum lengths, and unique occurrences for data values.

GRAPHICAL INTERFACE, DRAG-N-DROP MOTIF
XMLMap generation is accomplished via a familiar drag-n-drop motif. Data elements desired to be extracted from the XML sample are dragged directly from the input tree and dropped onto the XMLMap target tree. The target tree is organized, not remarkably, in table column precedence. Multiple table extractions are supported. Atlas also provides control dialogues that adjust the properties of the extracted columns. You can manipulate the detected data type, length, format, informat, as well as provide enumeration constraints on the column.

THE GHOSTS OF SXLE FUTURE (9.1)

SXLE
Scheduled for mass shipment, SAS 9.1 is the next evolutionary

step for our XML support in the field. We have augmented our success with new functionality and updated familiar tools with requested enhancements.

MS-ACCESS SUPPORT

Market presence is always a driving force behind any movement, whether it be a change of headache remedy or a preferred XML markup standard[4]. The newest native support XMLtype is the markup standard emitted by the Microsoft Access data base product. This markup is quite similar to the existing GENERIC XMLtype. MS-Access also supports inclusion of XML Schema as either embedded or separate file content.

NO SCHEMA OR SCHEMA OUTBOARD

To process MS-Access generated markup, on the LIBNAME statement specify the MSAccess option.

```
LIBNAME myxml xml 'path-to-my-xml-data'
xmltype='MSAccess' ;
```

If there is no embedded XML Schema content, the processing is exactly as with XMLtype=GENERIC. A first pass of the data is made to glean metadata, and data is extracted during the second pass. Note: Without the XML Schema content the results will be similar to, but may not match exactly, the database schema information.

SCHEMA EMBEDDED

With XML Schema content embedded in the file, the metadata is read directly from the information contained in the file. Data types are recognized as defined by the XML Schema – Datatypes specification. See the table below for supported types and type coercion.

XMLMETA OPTION VALUES CHANGED

Prior incarnations of the XMLmeta option enumeration values used somewhat cryptic forms like "yes", "full", "no", etc. We have made an effort in SAS 9.1 to use more meaningful names for these values, and match those in use in web applications. In the future, we will refer only to these new enumerations.

XMLMeta=DATA produces only data content in the XML output file. XMLMeta=SCHEMA produces only XML Schema content in the XML output file. So, one might safely assume that, XMLMeta=SCHEMADATA produces both XML Schema and data content in the XML output file(s). These are logically the successors of the values NONE, ONLY[5], and FULL, respectively.

XMLSCHEMA OPTION RE-INTRODUCED

The XMLSchema option has been re-introduced in SAS 9.1, and now specifies either a file reference or file path where generated XML Schema content is to reside. Initially, this option is restricted to output only.

With upcoming future releases, this is the location for references to generalized XML Schema content as support for the XML Schema specification is added to the engine and expanded.

PARSER

Parsing issues continue to arise as applications push the envelope and we have responded in several arenas of contention.

ALLOW "DIRTY" XML CHARACTERS

"True" XML is more than tossing a few angle brackets around existing data and calling it good. But, in direct conflict with the written specification, we see more and more occurrences of non-escaped character sequences in open XML fields. This sort of *faux pas* causes a conforming parser to stop processing at the

point of the offending character. We realize that this can be a frustrating experience for those tasked with processing third-party generated, non-conforming streams.

We now support a "relaxed" mode of processing that allows un-escaped special characters to be processed without error when they would not normally be accepted. The angle brackets in open text are still forbidden, but the other special characters (apostrophe, double quote, and ampersand) are processor pacified. On the LIBNAME statement, specify the XMLProcess option

```
LIBNAME myxml xml 'path-to-my-xml-data'
XMLProcess=Conform;
```

The option takes values of either Conform or Relax. The default is XMLProcess=Conform.

FASTER, SMALLER, SMARTER

Many improvements in the parser itself have streamlined the processing of XML inputs. Users should (continue to) expect good performance and resource usage with even the largest of streams.

XMLMAP

In keeping with a strict version control on syntax levels, SAS 9.1 introduces a new version "1.2" syntax level.

DATATYPE ENUMERATION CHANGE

Fundamentally compatible with version "1.1" the content of the DATATYPE enumeration has been changed to conform directly to the XML Schema – Datatypes specification. The old values are NOT accepted by the "1.2" version. Please refer to the table below

Schema type	Version 1.2	Version 1.0 Version 1.1
string	string	STRING
integer	integer	INT
float	double	FLOAT
double	double	FLOAT
dateTime	dateTime	DT-8601
time	time	TIM-8601
date	date	DAT-8601

Previous :
```
<DATATYPE>DT-8601</DATATYPE>
```

New in 1.2:
```
<DATATYPE>dateTime</DATATYPE>
```

THE SAS XML MAPPER (ATLAS)

Atlas was an XMLMap support tool introduced as beta software in SAS 9. The application has now been officially dubbed the SAS XML Mapper, and is production status in SAS 9.1.

SCHEMA SUPPORT

The production application now supports extraction from an XML-Schema, as well as, from more traditional XML data markup.
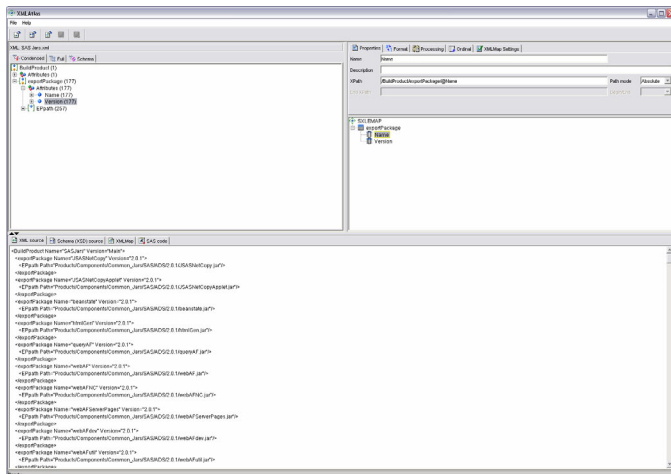
Element organization and content is displayed in a tree format as described by the schema definition.

BETTER TYPE RECOGNITION
The data type recognition has been expanded to include SAS-centric dates/times/datetimes in addition to ISO-standard forms. Recognition now parallels SXLE GENERIC mode operation.

TABBED DISPLAY
The application also now displays the source and generation text panes in a tabbed display format.

SAS DATA STEP XML PARSER OBJECT
Also new in SAS 9.1 is the Data Step XML Parser Object. This extension to the SAS Data Step permits direct access to the same XML parsing mechanics used by SXLE, in a non-engine environment.

SAX MODEL
The initial release of the object offers a SAX (Simple API for XML) model parser interface, and supplies the fundamental methods expected in a reference implementation.

CONCLUSION
The promise of XML uniting all IT departments under a common umbrella is still a pipedream. In this presentation we've taken a look at new technology to help you leverage the power of the SAS system and the new XML-based environments, specifically the new enhancements to the XML LIBNAME engine which extend functionality beyond the specific supported forms.

Examples have shown only basic code and related contextual fragments. For more in-depth coverage, please refer to the release documentation. There is also a SAS Theater Session being conducted during this SUGI which will deal with SXLE issues and examples more exhaustively.

FOOTNOTES
1   xmlengine@sas.com
2   DTD's are obsolete and XML Schema will be the only fully supported form in future releases. These tagsets produce output-only rendering support for application compatibility and are not intended for input to SXLE. There are many tools are available to convert an existing DTD to XML Schema notation.
3   http://www.w3.org/TR/REC-xml#sec-white-space
4   sometimes the two go hand-in-hand
5   ONLY was never formally documented prior, as it was an internal use option only. Use of ONLY in prior versions may lead to undesirable results. Offer void where prohibited. Your mileage may vary. See your dealer to see if you qualify.

REFERENCES
Resources: http://www.sas.com/rnd/base/index-xml-resources.html

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact me and the XML team at:

Anthony Friebel
XML Development Manager, SXLE architect
SAS Institute, Inc.
One SAS Circle
Cary, NC 27513
Email: XMLEngine@sas.com
Web: http://www.sas.com

COPYRIGHT INFORMATION

APPENDIX A.

XMLMAP SYNTAX ELEMENTS

The following are the XML tag names for the XMLMap syntax elements arranged by SXLE version. The tag names are listed in the order in which you would typically code them in your XMLMap file:

v0.9944	v1.0	v1.1	v1.2
XMLMAP	SXLEMAP	SXLEMAP	SXLEMAP
	version=	version=	version=
		name=	name=
		description=	description=
TABLE	TABLE	TABLE	TABLE
NAME			
	name=	name=	name=
TABLE_LABEL	TABLE_LABEL		
		TABLE-DESCRIPTION	TABLE-DESCRIPTION
TABLE_XPATH	TABLE_XPATH	TABLE-PATH	TABLE-PATH
		syntax=	syntax=
	TABLE_END_XPATH	TABLE-END-PATH	TABLE-END-PATH
		syntax=	syntax=
		beginend=	beginend=

COLUMN	COLUMN	COLUMN	COLUMN
NAME			
	name=	name=	name=
	retain=	retain=	retain=
	replace=		
		ordinal=	ordinal=
LABEL	LABEL		
		DESCRIPTION	DESCRIPTION
XPATH	XPATH	PATH	PATH
		syntax=	syntax=

COLUMN	COLUMN	COLUMN	COLUMN
		INCREMENT-PATH	INCREMENT-PATH
		syntax=	syntax=
		beginend=	beginend=
		RESET-PATH	RESET-PATH
		syntax=	syntax=
		beginend=	beginend=
XMLTYPE	DATATYPE	DATATYPE	DATATYPE
SASTYPE	TYPE	TYPE	TYPE
LENGTH	LENGTH	LENGTH	LENGTH
	FORMAT	FORMAT	FORMAT
	width=	width=	width=
	ndec=	ndec=	ndec=
	INFORMAT	INFORMAT	INFORMAT
	width=	width=	width=
	ndec=	ndec=	ndec=
	ENUM	ENUM	ENUM
	VALUE	VALUE	VALUE
	DEFAULT	DEFAULT	DEFAULT

The v0.9944 level syntax was not officially documented and was fairly soon after supplanted by the introduction of v1.0 syntax.