

## Paper 25-28

## XML in the DATA Step

Michael Palmer, Zurich Biostatistics, Inc., Morristown, New Jersey

**ABSTRACT**

This paper discusses a DATA-step method to import, export, and transform user-defined XML vocabularies. It contrasts with SAS® Institute (SI) XML tools in that it uses a single, uniform methodology to import, export, and transform a broad class of user-defined XML vocabularies. This gives more control to SAS users than the SI tools offer.

From the point of view of a SAS user, XML is trouble for three reasons: it's hierarchical, it's all text, and all content is explicitly named. A DATA-step methodology for working with XML should (1) flatten the hierarchical XML into the rows-and-columns of SAS datasets without losing the information needed to recreate the hierarchy and (2) replace with numbers the unpredictably long and complex text strings that describe paths to data in XML.

A straightforward algorithm for numbering levels of depth in an XML hierarchy and for numbering siblings at a level satisfies these conditions but has a problem. The indexes that it produces will not be invariant to the way an XML instance happens to be put together. For invariant indexing, the algorithm has to be tied to the XML vocabulary for the instance, not to the instance itself. The presentation will discuss how this invariant indexing has been done. The methodology supports highly generic programming, and relies on two macros, %importXML and %exportXML.

The method has been in use successfully and supports complex XML vocabularies such as the pharmaceutical industry's CDISC XML standard for clinical data. It has scaled to XML files of several hundred thousand records.

**INTRODUCTION**

XML is a hierarchical, text-based format consisting of named content. These three characteristics make it difficult to work with in the row-and-column-oriented DATA-step and file structure in SAS. The paper discusses a DATA-step friendly way to work with user-defined XML vocabularies. The method supports the import of XML into SAS, the export of XML from SAS, and the processing and transformation of XML in the DATA-step.

SAS Institute offerings for working with XML include a LIBNAME engine for importing and exporting some fairly simple types of XML, including some common industry forms. The XML Map option of the LIBNAME Engine can import a broader class of XML directly to SAS datasets. A graphical user interface tool is also provided for assisting in creation of XMLMaps from raw XML data in a familiar drag-and-drop motif. ODS can automatically export a SAS-defined type of XML and some common industry forms. With customized tagset extension programming, ODS may export some user-defined types of XML. Version 9 also contains broader support for parsing XML in the DATA-step.

What's missing from the SAS Institute XML support are,

- a unified way to import and export user-defined XML vocabularies in the DATA-step, and
- a way to perform basic XML processing such as transforming one XML instance into another instance in the same XML vocabulary or in a different vocabulary.

The second bulleted point is functionality similar to XSLT but done in the DATA-step using familiar DATA-step programming techniques.

The methodology discussed here contrasts with SAS Institute XML tools in several ways.

1. It works with a very broad class of user-defined XML vocabularies.
2. It uses one uniform methodology for the import, export, and transformation of any instance of any data-centric XML vocabulary.
3. XML work all takes place in the base SAS DATA-step. The programming techniques necessary to implement the method are familiar to even beginning SAS programmers.

SI's XML tools do not have these characteristics.

The method has been in use successfully for several years and does support real life, complex, consortium-developed XML vocabularies such as the pharmaceutical industry's CDISC XML standard for clinical study data. The method has scaled successfully to XML files of several hundred thousand records.

**WHAT'S THE PROBLEM WITH XML AND SAS?**

XML is *hierarchical*, unlike the typical SAS dataset. In the typical SAS setting, data exist in fields on records in datasets. In a given dataset, every record has precisely the same fields with precisely the same attributes. One relates a data item to another data item by the fact that they share, or do not share, key variables and key variable values. In XML, a data item relates to other data items by the ancestors, descendants, and siblings that they share. XML is hierarchical so a data item inherits information from its ancestors, shares information with its siblings, and passes on information to its descendants. To identify a data item, one has to literally traverse all of its ancestors. This traversal creates a path through the XML file. By contrast, in the typical SAS dataset, one identifies a data item by looking at key fields on the same record.

XML is *text*. All XML is text, accessible in a simple text editor. In SAS datasets, by contrast, fields can be text or numeric and, despite the rich set of text-processing functions in SAS, numeric data is easier to process than text data in SAS. In addition, proprietary tools such as SAS Viewer or the base SAS product are necessary to access data and attribute information in a SAS dataset.

XML consists of *named content*. The name of each item in an XML file is written completely in text when that element's scope begins and written out again completely when that element's scope ends. The sequence of items in that scope is predictable to some extent, but not fixed like it is for a typical SAS dataset. In addition, field attributes, such as field name, are not explicit in SAS but are stored separately from the data itself.

From the point of view of a SAS user, XML is trouble for these three reasons: it's *hierarchical*, it's all *text*, and all *content is explicitly named*.

Despite these complications, SAS users would like to be able to import XML, export XML, and work with XML in the DATA-step as easily as they work with other data originating in other formats.

**A SUCCESSFUL METHODOLOGY**

A successful methodology for working with XML in the DATA-step would include the following three features.

One, the methodology should flatten the hierarchical XML structure into the row-and-column structure of SAS datasets but without losing the information needed to recreate the hierarchy. That is, the

ancestor-sibling-descendent structure in native XML should be indexed into a flat representation.

Two, the methodology should replace the unpredictably long and complex text strings that describe paths to data items in XML with numbers. Long and complex text strings are cumbersome to work with in the SAS DATA-step. Numbers are easy to work with.

Three, the methodology should fit the numerically indexed content into a regular row-and-column SAS dataset.

A straightforward algorithm for numbering levels of depth in an XML hierarchy and for numbering siblings at a given level satisfies these three conditions but has a problem. The problem is that, unlike flat files, XML instances for a given XML vocabulary don't necessarily have identical, or even similar structures. They can, and often have, very heterogeneous structures. It's even possible to have the identical content, that should end up in identical data structures in SAS, in very different XML structures from the same XML vocabulary. This situation makes the straightforward algorithm unreliable because the indexes that it produces will not be invariant to the way an XML instance happens to be put together. To make the indexing invariant, it has to be tied to the structure of the XML vocabulary for the instance, not to the instance itself.

The presentation will discuss how this invariant indexing has been done. The methodology supports highly generic programming and it has been implemented to take advantage of that support. In the author's implementation, there are two macros, %importXML and %exportXML. These are available for free from the author.

With these two macros, XML-formatted data from a very broad class of user-defined XML vocabularies can be imported, exported, and processed in SAS without having to leave the base SAS environment. This contrasts with the SAS Institute's methodology which directs SAS users to use XSLT outside of SAS for XML transformations. A significant advantage of %importXML and %exportXML over SI's XML tools is that they can always import and export user-defined XML without programming. They work very much like XML analogs to INPUT and OUTPUT statements in a DATA-step. This is a significant simplification of XML processing in SAS.

## XML IN THE DATA-STEP

The XML fragment below will be used to illustrate the indexing algorithm. XML statements begin with a start tag of the form <TagName> and end with a tag of the form </TagName>. The scope of a tag may completely include other tags and content: <Tag00><Tag01>Information</Tag01></Tag00>. In addition, a tag may have one or more attributes in its scope: <Tag00 Attribute00="Stuff">. Tags that do not contain other tags or content may be written as empty tags: <Tag02 Attribute01="More stuff"/>.

The XML fragment below has three tag names. <SubjectData> is in the scope of <ODM> and <ItemData> is in the scope of <SubjectData>. <ItemData> has two attributes with data: ItemOID and Value.

```
<ODM >
  <SubjectData>
    <ItemData ItemOID="PT" Value="P002"/>
  </SubjectData>
</ODM>;
```

To index this fragment, ODM is at the root, or ultimate, level so it has a value of 1 for the root level indexing variable. SubjectData is in the scope of ODM so it inherits a root level indexing variable of 1, and, in addition, since SubjectData is the first tag one level inside the root level, it has a second index variable with a value of 1. ItemData, inside the scope of SubjectData, inherits its index variables and gets one new one for itself. ItemOID and Value are attributes but, for

purposes of indexing, they are treated just like tags inside the scope of ItemData. The XML fragment below shows the complete indexing.

```
1      <ODM >
1 1    <SubjectData >
1 1 1  <ItemData
1 1 1 1 ItemOID="PT"
1 1 1 2 Value="P002"/>
      </SubjectData >
</ODM>
```

Looking at the indexed XML, it's straightforward to see how it could be stored in a SAS dataset, and Figure 1 shows this fragment imported into a SAS dataset.

## SIMPLE PROGRAMMING TECHNIQUES WORK

The SAS DATA-step code below shows how this XML imported into a SAS dataset can be processed using familiar DATA-step techniques and put directly into two fields, NAME and VALUE, in a SAS dataset.

```
/*<ItemData ItemOID="" />*/
if ID01=1 and ID02=1 and ID03=1 and ID04=1
then NAME=trim(left(content0));

/*<ItemData Value="" />*/
if ID01=1 and ID02=1 and ID03=1 and ID04=2
then VALUE=trim(left(content0));

output;
run;
```

In the next XML fragment below, a second <ItemData> element is added and indexed.

```
1      <ODM >
1 1    <SubjectData >
1 1 1  <ItemData
1 1 1 1 ItemOID="PT"
1 1 1 2 Value="P002"/>
1 1 2  <ItemData
1 1 2 1 ItemOID="PT"
1 1 2 2 Value="P002"/>
      </SubjectData >
</ODM>
```

In order to process this new ItemData, the SAS code would also have to be modified to pickup the new 1 1 2 element. Obviously, a method of handling XML in SAS that required custom code to match the XML instance would not be useful.

The indexing algorithm has to recognize repeating XML structures and give them identical indexes, as below.

```
1      <ODM >
1 1    <SubjectData >
1 1 1  <ItemData
1 1 1 1 ItemOID="PT"
1 1 1 2 Value="P002"/>
1 1 1  <ItemData
1 1 1 1 ItemOID="PT"
1 1 1 2 Value="P002"/>
      </SubjectData >
</ODM>
```

With both ItemData elements indexed the same, the SAS code sample becomes more generic and practical.

## A GENERAL SOLUTION

A general solution to the need for an indexing algorithm that is invariant to the XML instance, but depends on the underlying structure of the XML vocabulary is called, in Zurich Biostatistics' Tekoa Technology<sup>sm</sup>, a canonical document (a candoc). The candoc for the XML fragment under discussion is, in indexed form,

```
1      <ODM >
1 1    <SubjectData >
```

```

1 1 1      <ItemData
1 1 1 1    ItemOID=" "
1 1 1 2    Value=" " />
      </SubjectData >
</ODM>

```

The candoc contains no content but defines the permitted tag names, attribute names, and relationships between all tags and attributes. It's a template to guide the import and export of an XML vocabulary. Using the candoc, every ItemOID with a path through the XML of <ODM><SubjectData><ItemData> will have exactly the same ID: 1 1 1 1, and will be processed correctly in the SAS code sample above. In other words, the code sample will work very much like a traditional INPUT statement in a DATA-step.

This brings the SAS programmer control of XML processing in the DATA-step but *does not require knowledge of XPath*. Since the %importXML tool *automatically indexes* any canonical document and *automatically indexes* any XML instance, the SAS programmer can use familiar DATA-step programming techniques, as shown above, to work with XML.

Figure 2 shows the canonical document for this XML imported into a SAS dataset. From here, it is used to guide import and export of XML.

Figure 3 shows the XML fragment with two ItemData statements imported to a SAS dataset according to the candoc. From this dataset, the XML is ready for DATA-step processing using familiar techniques.

#### A 100% BASE SAS SOLUTION FOR ALL XML (AND NO ODS!)

Candocs guide the import of XML to SAS and they guide the export of XML from SAS. ZBI's %exportXML tool uses a candoc to write XML from the kind of XML SAS dataset discussed in this paper. The *XML export does not require ODS or Java or perl*. Export, like import, is done completely in the base SAS DATA-step using familiar techniques.

Candocs, since they are completely under the control of the user, allow anyone who is comfortable with DATA-step programming to import and export any data-centric XML vocabulary.

#### XSLT-TYPE TRANSFORMATIONS

XSLT-type transformations can be done in the DATA-step using this indexing approach. To transform, the index ID values are changed with DATA-step programming statements. Since the index ID values identify paths in XML, one instance can be transformed to another by changing the ID values and exporting the resulting XML with the appropriate candoc.

#### SUMMARY

To summarize, XML's hierarchical, text-based, named content nature makes it cumbersome to import, export, and transform in the DATA-step. A sleek way of handling XML in the DATA-step involves flattening the hierarchical representation into a numerically indexed representation that presents nothing new to the SAS programmer, unlike native XML. This method has been implemented in a highly generic way, used with real life XML, and scaled up to production-size XML files of several hundred thousand records.

#### CONCLUSION

Despite unfortunate experiences that savvy SAS programmers have had working with XML in the SAS DATA-step, XML and SAS can coexist nicely, for import of XML into SAS, export of XML from SAS, and transformation of one XML instance into another. This is true for any data-centric XML that a user may encounter.

#### REFERENCES

Zurich Biostatistics, Inc. presentations archive:  
[www.zbi.net/NewFiles/leadership.html](http://www.zbi.net/NewFiles/leadership.html)

SAS Institute XML web pages:  
[www.sas.com/rnd/base/index-xml-resources.html](http://www.sas.com/rnd/base/index-xml-resources.html)

#### CONTACT INFORMATION

Your comments and questions are valued and encouraged. Additionally, free copies of %importXML and %exportXML are available from the author.

Contact the author at:

Michael Palmer  
 Zurich Biostatistics, Inc.  
 45 Park Place, So., PMB 178  
 Morristown, NJ 07960  
 Phone: 973-277-9034  
 Email: [mcpalmer@zbi.net](mailto:mcpalmer@zbi.net)  
 Web: [www.zbi.net](http://www.zbi.net)

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.

Tekoa Technology<sup>sm</sup> is a service mark of Zurich Biostatistics, Inc.

Other brand and product names are trademarks of their respective companies.

## FIGURES

Figure 1. XML instance in SAS dataset with ID variables.

The screenshot shows the SAS software interface with a dataset view titled 'VIEWTABLE: Stage 1 dataset'. The dataset contains five rows of XML instances and their corresponding ID variables. The columns are DOC\_CODE, CONTENTO, DOC\_D, ID0, ID1, ID2, and ID3. The data is as follows:

	DOC_CODE	CONTENTO	DOC_D	ID0	ID1	ID2	ID3
1	<ODM>		1	1	.	.	.
2	<SubjectData>		2	1	1	.	.
3	<ItemData>		3	1	1	1	.
4	<ItemOID ATT="1">	PT	4	1	1	1	1
5	<Value ATT="1">	P002	5	1	1	1	2

The interface also shows a taskbar with several open windows: Output - (Unti..., Log - (Unti..., Editor - Untitl..., ImportCDISC..., Explorer, and VIEWTAB... The status bar at the bottom indicates the current directory is C:\WINNT\Profiles\A.

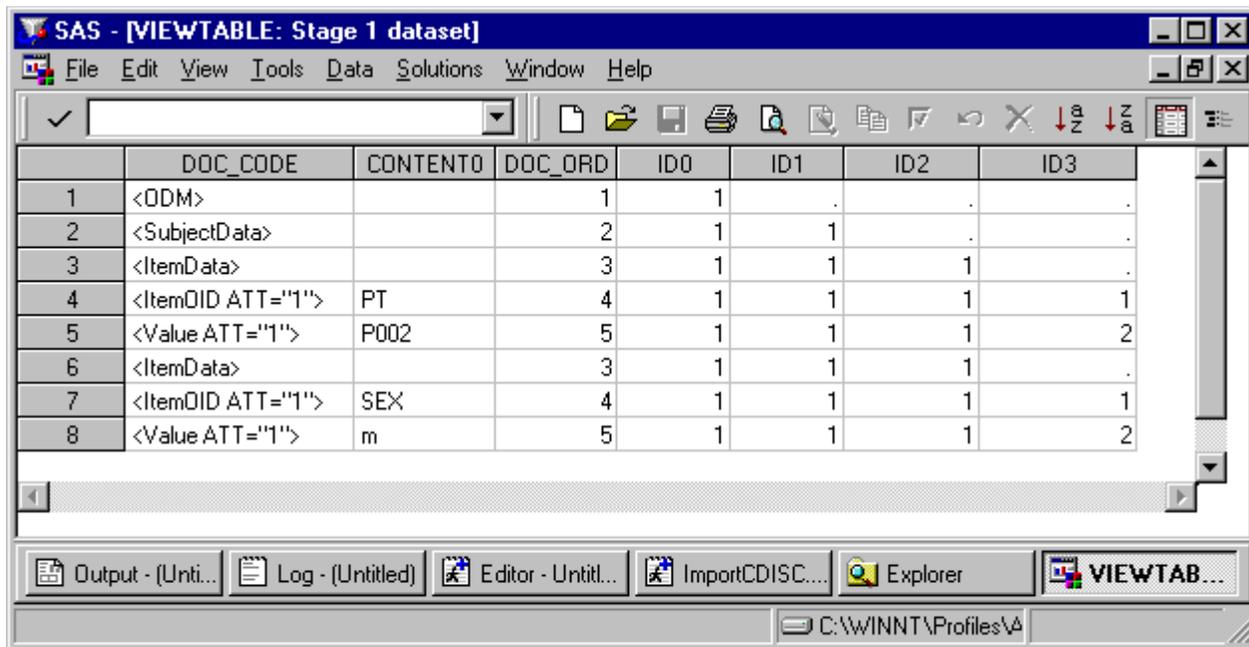
Figure 2. Canonical XML document.

The screenshot shows the same SAS software interface as Figure 1, but with a different dataset view titled 'VIEWTABLE: Stage 1 dataset'. The dataset contains five rows of XML instances and their corresponding ID variables. The columns are DOC\_CODE, CONTENTO, DOC\_ORD, ID0, ID1, ID2, and ID3. The data is as follows:

	DOC_CODE	CONTENTO	DOC_ORD	ID0	ID1	ID2	ID3
1	<ODM>		1	1	.	.	.
2	<SubjectData>		2	1	1	.	.
3	<ItemData>		3	1	1	1	.
4	<ItemOID ATT="1">		4	1	1	1	1
5	<Value ATT="1">		5	1	1	1	2

The interface also shows the same taskbar and status bar as Figure 1.

Figure 3. XML instance in SAS dataset with ID variables. Note that the repeating <ItemData> group in the XML has repeating Ids.



The screenshot shows the SAS software interface with a dataset view titled "VIEWTABLE: Stage 1 dataset". The dataset contains 8 rows of data, each representing an XML instance. The columns are labeled DOC\_CODE, CONTENTO, DOC\_ORD, ID0, ID1, ID2, and ID3. The data is as follows:

	DOC_CODE	CONTENTO	DOC_ORD	ID0	ID1	ID2	ID3
1	<ODM>		1	1	.	.	.
2	<SubjectData>		2	1	1	.	.
3	<ItemData>		3	1	1	1	.
4	<ItemOID ATT="1">	PT	4	1	1	1	1
5	<Value ATT="1">	P002	5	1	1	1	2
6	<ItemData>		3	1	1	1	.
7	<ItemOID ATT="1">	SEX	4	1	1	1	1
8	<Value ATT="1">	m	5	1	1	1	2

The interface also shows a taskbar with several open windows: Output - (Unti..., Log - (Untitled), Editor - Untit..., ImportCDISC..., Explorer, and VIEWTAB... The system tray shows the path C:\WINNT\Profiles\A.