A Method to Import, Process, and Export Arbitrary XML^{TM} Files with SAS$^{®}$

Michael C. Palmer, Zurich Biostatistics, Inc., Morristown, NJ
Cecilia A. Hale, Zurich Biostatistics, Inc., Morristown, NJ

ABSTRACT

The release of the FDA and industry-supported CDISC XML clinical data standard will make SAS's XML capabilities increasingly important. XML-formatted data are text with the hierarchical, tagged structure of markup languages, not the row and column style of SAS datasets. SAS has some experimental XML capabilities but it cannot import, process, and export arbitrary XML vocabularies as XML.

This paper presents a method to import, process, and export arbitrary, data-centric XML files.

The method consists of an indexing algorithm, construction of a canonical instance, and processing of populated instances. Populated instances are put into a canonical form, and the algorithm is applied. The resulting indexed instance is pruned to just records that contain data and stored as a SAS dataset with the index as a key, shorn of the markup structure that is hard to work with in SAS. The pruned instance can be mapped into SAS datasets and processed with the index, preserving the XML structure.

INTRODUCTION

Historically, the processing of clinical data from initial collection to submission documents has depended on proprietary or ad hoc standards and specifications. New and continuing forces in the pharmaceutical industry including mergers, acquisitions, out-sourcing, use of central clinical laboratories, international regulatory harmonization, and the FDA's mandate to accept electronic submissions have focused attention, sometimes unwanted, on the time, labor, and expense required by the historical practices.

Recently, the pharmaceutical industry, in the form of CDISC (Clinical Data Interchange Standards Consortium, Inc.) has embraced XML as an enabling technology to help streamline and improve the industry's historical practices. The bottom line for those of us working with SAS is that we're going to be faced with the necessity of importing, processing, and exporting XML.

LEGACY PRACTICES

SAS does not have the capability to import, process as XML, and export arbitrary XML-formatted data. Beginning with version 8, SAS has limited capabilities to import and export XML using an XML engine in the LIBNAME statement. ODS offers some XML export capabilities for SAS output, but not for data per se. DATA-step programming, particularly text processing capabilities, can be used because XML files are always just plain text.

On the XML side, several approaches exist for database programming and could be evaluated for implementation in SAS (See reference 1).

NEW TECHNOLOGY

The technology presented in this paper translates an instance of XML-formatted data into a flat file representation which can be processed alone or with other similarly translated data and then translated back to valid and well-formed XML of the same vocabulary. This work has been developed as part of Zurich Biostatistics' Tekoa Technology^{SM}.

XML (eXtensible Markup Language) is a non-proprietary, platform-independent meta-language for hierarchically structuring information. XML vocabularies, generally defined by Document Type Definitions (DTDs) or schemas (see reference 2), use the meta-language to define elements and the rules on how to use them. An XML instance is a text file of data conforming to a specific vocabulary.

XML is extensible in the sense that anyone with an understanding of the rules of XML can make up their own XML vocabulary. XML is a markup language because it provides the means to describe documents of data. The contents of XML documents can go into a database with relative ease or the content of a database can go into XML documents.

The World Wide Web Consortium (W3C) guided the creation of XML in a joint effort of many industry and academic contributors. The XML standard was released in February 1998. The W3C is a non-profit consortium made up primarily of corporate and other organizations with an interest in the development of the World Wide Web.

The technology discussed in this paper has been implemented in part for the OASIS table model in Tekoa Technology table automation software (see reference 3). This paper is a generalization of that earlier work. It falls into the "elements as fields" strategy discussed by Quin (reference 1).

The method described here is essentially a variation of the "Edge" approach described by Tian et. al. (see reference 4). A significant difference is that they consider the case of DTD-less XML fragments, and we consider the case of DTD-defined XML and attempt to exploit the existence of the DTD with our canonical document.

THE NEED

Data-centric XML (see reference 5) is driving standards development in pharma and many, many other industries. Each standard is a separate and distinct vocabulary, often rich enough to support complex data like clinical studies but leading to tremendous variability in file structure when compared to legacy flat files. SAS professionals will increasingly be called upon to import, export, and process XML-formatted data that conform to these emerging standards. The technology discussed in this paper is a general way to handle these vocabularies.

IMPORT OF XML

When compared to the traditional, row and column structure, XML's hierarchical, markup language file structure guided by DTDs looks structureless. Data arrive in a text stream of named values. The hierarchical relationships between the named values must be ascertained by examining the markup language that accompanies the data itself. Translating this to the typical SAS dataset requires converting the hierarchical relationships and the data to row-column structure.

PROCESSING XML-BASED DATA

The hierarchical relationships among data elements in XML-formatted data must be used to identify data types, such as blood pressure data for patient x on study day y. In practical terms, programs cannot simply, for example, merge datasets to bring together demographic and efficacy data because the keys for the merge exist in a tree, not in records in a flat file. Whatever processing that is done must preserve the hierarchical relationships so that the data can be exported as XML (that is, "round tripping"). This means that combining data from different sources requires a common instance-independent way of defining the hierarchy.

EXPORT OF XML

The preservation of the hierarchical relationships during processing makes it possible to export valid and well-formed XML documents.

IMPLEMENTATION

The initial, limited implementation of Tekoa Technology for the OASIS table model was completely in base SAS version 8.

COMPONENTS

In the terminology of Tekoa Technology, a *populated instance* is translated into a *pruned instance*, and vice versa. The translation is mediated by a particular representation of the DTD. In Tekoa Technology terminology, this representation is called the *canonical document*.

CANONICAL DOCUMENT

A canonical document is, in effect, an empty record for a data-centric DTD. Like an empty record for a row-column file, it defines a data structure for import, export, and processing. The canonical document exists as XML.

In the canonical document, each element type and attribute in the DTD, or in a well-defined subset of the DTD, appears in the hierarchy that the DTD defines. To make the representation of attributes canonical, they appear in lexicographic order. A canonical document may be built for a subset of a DTD as long as the resulting instances are valid for the DTD.

A canonical document includes a multi-field index for each element and attribute, and the index reflects the element's or attribute's place in the data hierarchy.

Among the implementation choices not discussed here but important in any implementation are wrapping content in dummy elements, converting null element types ("empty" elements in XML terminology) to their equivalent non-null form, and converting attributes to elements.

In a way similar to the way an empty record serves as a data layout template for flat files, a canonical document is a template for XML data files. Every element and attribute in the canonical document must appear in every populated instance of the canonical document, even if it has no content or value in the instance, i.e., is null. This is necessary for the indexing to be invariant to instance. This invariance must be maintained so that data from different instances with the same index will be of the same type.

As mentioned above, this approach has been implemented in a table automation project for the OASIS table model and it should work with a broader class of data-centric DTDs.

POPULATED INSTANCE

A populated instance is simply a canonical document populated with data, where the data exist, and null where it does not.

PRUNED INSTANCE

The pruned instance is a flat file representation of the populated instance. One transforms a populated instance into a pruned instance by removing the markup language, leaving the index and content, and adding a document order counter for each indexed element of content. Since the index is often shorter than the markup language it replaces, this can result in substantially smaller files than the original XML.

INDEXING ALGORITHM

The indexing algorithm is a way to represent with numbers a nested data structure.

To generate the index conceptually, the canonical document is laid out with each element indented if it is nested in the previous element and otherwise not indented. Attributes are treated as if they are indented in the element that contains them. The index has one field per indentation level, and the value in a field at any point is the number of elements that have occurred in that field since the last element that contained the field. The index for an element is the concatenation of all the fields. A null value for a field occurs when an element contains elements corresponding to that field, or is at the same indentation level as an element that contains elements corresponding to that field.

The relationship between any two records with the same canonical document can be determined by comparing their index values. This is a powerful tool for subsetting, merging, and otherwise processing the flat file representation of the XML because it provides a very compact summary of a data value's relationship to any other data value.

EXAMPLE

The example is part of a sample of clinical data that has been distributed with the CDISC Operational Data Model (ODM) DTD (see reference 6). The canonical document for the example is the subset of the full DTD that just describes the sample.

XML PRIMER

First and foremost, XML is just text. From the perspective of SAS and SAS programming, one must always keep in mind this simplifying fact: each and every XML file, whether imported or exported, is just text.

The XML markup language has elements, attributes, and content structured hierarchically, elements nested in elements, attributes nested in elements, and content nested in elements. For example,

```
<PatientData Patientid="B00-2136-001"
EnrollDate="02-Jun-2000" Gender="female"
Initials="AMH">
  <StudyEventData>
    <StudyEventKey StudyEventID="VISIT_1">
    </StudyEventKey>
    <ItemData>
      <ItemKey ItemID="PT">
      </ItemKey>
      <Value>P001</Value>
    </ItemData>
    <SignatureDefRef SignatureDefID="001" />
  </StudyEventData>
</PatientData>
```

Element tags begin and end with left and right carets, < and >. Each element tag has a name in its start tag, <StudyEventKey> for example, and may have attributes, <TagName Att="VISIT_1">. The end of an element's scope is delimited with an end tag, </TagName> for instance. Some elements are null, they have no scope but may have attributes, and their tags have the form <TagName att="Hi!" />. Attributes can appear in any order inside an element. Element order must conform to the DTD-defined syntax.

DTDs define XML vocabularies by defining element and attribute names and relationships, among other things. An XML instance that conforms to both the XML standard and to a specific DTD is termed, "well-formed and valid."

As the XML fragment above shows, data can come as an attribute or as content delimited by tags. For data-centric XML, the distinction between attribute and element as the vehicle for a data element is sometimes arbitrary.

Users of vocabularies supply the content and semantic understanding that make a particular instance meaningful.

POPULATED INSTANCE

Clinical data from a CDISC ODM sample is below. A SAS programmer facing this text stream for the first time, rather than the familiar row and column structure, might justifiably feel that XML adds complexity to data processing.

The pruned instance, on the other hand, has the familiar row and column look and also sufficient information on element order and hierarchy to combine the data in this instance with data in other instances, map the data to a standard clinical database, and export the data in XML in the ODM vocabulary. The point of this paper is to present a method to translate the populated XML instance into the pruned instance.

```
<PatientData Patientid="B00-2136-001"
EnrollDate="02-Jun-2000" Gender="female"
Initials="AMH" DOB="07/16/1947" RandDate="05-
Jun-2000">
<PatientKey>P001</PatientKey>
<!--StudyEventData element for Visit 1 for
this patient-->
<StudyEventData><StudyEventKey
StudyEventID="VISIT_1"></StudyEventKey>
<!--SigningUnitData element for the Demog and
Vitals form collected on Visit 1-->
<SigningUnitData><SigningUnitKey
SigningUnitID="PAGE_1"></SigningUnitKey>
<!--ItemGroupData element for the
Demographics part of the form-->
<ItemGroupData><ItemGroupKey
ItemGroupID="DEMOG"></ItemGroupKey>
<ItemData><ItemKey
ItemID="PT"></ItemKey><Value>P001</Value></It
emData>
<ItemData><ItemKey
ItemID="INITIALS"></ItemKey><Value>AMH</Value
></ItemData>
<ItemData><ItemKey
ItemID="GENDER"></ItemKey><Value>f</Value></I
temData>
<ItemData><ItemKey
ItemID="DOB"></ItemKey><Value>07/16/1947</Val
ue></ItemData>
<ItemData><ItemKey
ItemID="SPONSOR_PTID"></ItemKey><Value>B00-
2136-001</Value></ItemData>
<ItemData><ItemKey
ItemID="WEIGHT_LB"></ItemKey><Value>150.00</V
alue></ItemData>
<ItemData><ItemKey
ItemID="WEIGHT_KG"></ItemKey><Value>68.18</Va
lue></ItemData>
</ItemGroupData>
<!--ItemGroupData element for the Vitals part
of the form-->
<ItemGroupData><ItemGroupKey
ItemGroupID="VITALS"></ItemGroupKey>
<ItemData><ItemKey
ItemID="PT"></ItemKey><Value>P001</Value></It
emData>
<ItemData><ItemKey
ItemID="VISITNAME"></ItemKey><Value>Visit1</V
alue></ItemData>
<ItemData><ItemKey
ItemID="SBP"></ItemKey><Value>120</Value></It
emData>
<ItemData><ItemKey
ItemID="DBP"></ItemKey><Value>80</Value></Ite
mData>
<ItemData><ItemKey
ItemID="SPONSOR_PTID"></ItemKey><Value>B00-
2136-001</Value></ItemData>
<ItemData><ItemKey
ItemID="OCCUR_NUM"></ItemKey><Value>1</Value>
</ItemData>
</ItemGroupData>
<!--Electronic signature for this form-->
<Signature><UserRef
UserID="User.001"/><LocationRef
LocationID="Location.001"/>
<SignatureDefRef
SignatureDefID="SignatureDef.001"/>
<DateTimeStamp>7-June-2000
11:15:42</DateTimeStamp></Signature>
</SigningUnitData>
</StudyEventData>
</StudyEventData>
</PatientData>
```

PRUNED INSTANCE

The pruned instance includes all of the element content and attribute values in the XML instance as well as indices that indicate position in the canonical document hierarchy for each piece of content. In terms of the row and column structure of the pruned instance, the first value is the order in the document of the content. The values following the comma are the multi-field index, and the third value is the content itself. The document order counter does not start at "1" because this is a fragment of the sample data file.

```
108,1 5 1 1             07/16/1947
109,1 5 1 2             02-Jun-2000
110,1 5 1 3             female
111,1 5 1 4             AMH
112,1 5 1 5             B00-2136-001
113,1 5 1 6             05-Jun-2000
114,1 5 1 7             P001
115,1 5 1 8 1 1         VISIT_1
116,1 5 1 8 2 1 1       PAGE_1
117,1 5 1 8 2 2 1 1     DEMOG
118,1 5 1 8 2 2 2 1 1   PT
119,1 5 1 8 2 2 2 2     001
120,1 5 1 8 2 2 2 1 1   INITIALS
121,1 5 1 8 2 2 2 2     AMH
122,1 5 1 8 2 2 2 1 1   GENDER
123,1 5 1 8 2 2 2 2     f
124,1 5 1 8 2 2 2 1 1   DOB
125,1 5 1 8 2 2 2 2     07/16/1947
126,1 5 1 8 2 2 2 1 1   SPONSOR_PTID
127,1 5 1 8 2 2 2 2     B00-2136-001
128,1 5 1 8 2 2 2 1 1   WEIGHT_LB
129,1 5 1 8 2 2 2 2     150.00
130,1 5 1 8 2 2 2 1 1   WEIGHT_KG
131,1 5 1 8 2 2 2 2     68.18
132,1 5 1 8 2 2 1 1     VITALS
133,1 5 1 8 2 2 2 1 1   PT
134,1 5 1 8 2 2 2 2     P001
135,1 5 1 8 2 2 2 1 1   VISITNAME
136,1 5 1 8 2 2 2 2     Visit1
137,1 5 1 8 2 2 2 1 1   SBP
138,1 5 1 8 2 2 2 2     120
139,1 5 1 8 2 2 2 1 1   DBP
140,1 5 1 8 2 2 2 2     80
141,1 5 1 8 2 2 2 1 1   SPONSOR_PTID
142,1 5 1 8 2 2 2 2     B00-2136-001
143,1 5 1 8 2 2 2 1 1   OCCUR_NUM
144,1 5 1 8 2 2 2 2     1
145,1 5 1 8 2 3 1 1     User.001
146,1 5 1 8 2 3 2 1     Location.001
147,1 5 1 8 2 3 3 1     SignatureDef.001
148,1 5 1 8 2 3 4       7-June-2000 11:15:42
```

SAS DATASET

The pruned instance, unlike the original XML, has a two dimensional structure that can be readily read into a SAS dataset. Importantly, the document order counter and the index contain all of the data structure information of the original XML. They provide the means to move the imported data into a preexisting processing stream.

EXPORTING XML

Well-formed and valid XML is exported from the pruned instance using the canonical document. The part of the canonical document that corresponds to the data in the example is below.

```
<PatientData DOB= EnrollDate= Gender=
Initials= Patientid= RandDate= >
  <PatientKey> </PatientKey>
  <StudyEventData>
    <StudyEventKey StudyEventID= >
    </StudyEventKey>
    <SigningUnitData>
      <SigningUnitKey SigningUnitID= >
      </SigningUnitKey>
      <ItemGroupData>
        <ItemGroupKey ItemGroupID= >
        </ItemGroupKey>
        <ItemData>
          <ItemKey ItemID= >
          </ItemKey>
          <Value> </Value>
        </ItemData>
      </ItemGroupData>
      <Signature>
        <UserRef UserID= />
        <LocationRef LocationID= />
        <SignatureDefRef SignatureDefID= />
        <DateTimeStamp> </DateTimeStamp>
      </Signature>
    </SigningUnitData>
  </StudyEventData>
</PatientData>
```

The index values for the elements and attributes in the canonical document are below.

```
1 5 1              <PatientData>
1 5 1 1            <DOB>
1 5 1 2            <EnrollDate>
1 5 1 3            <Gender>
1 5 1 4            <Initials>
1 5 1 5            <Patientid>
1 5 1 6            <RandDate>
1 5 1 7            <PatientKey>
1 5 1 8            <StudyEventData>
1 5 1 8 1          <StudyEventKey
1 5 1 8 1 1        <StudyEventID>
1 5 1 8 2          <SigningUnitData>
1 5 1 8 2 1        <SigningUnitKey>
1 5 1 8 2 1 1      <SigningUnitID>
1 5 1 8 2 2        <ItemGroupData>
1 5 1 8 2 2 1      <ItemGroupKey>
1 5 1 8 2 2 1 1    <ItemGroupID>
1 5 1 8 2 2 2      <ItemData>
1 5 1 8 2 2 2 1    <ItemKey>
1 5 1 8 2 2 2 1 1  <ItemID>
1 5 1 8 2 2 2 2    <Value>
1 5 1 8 2 3        <Signature>
1 5 1 8 2 3 1      <UserRef>
1 5 1 8 2 3 1 1    <UserID>
1 5 1 8 2 3 2      <LocationRef>
1 5 1 8 2 3 2 1    <LocationID>
1 5 1 8 2 3 3      <SignatureDefRef>
1 5 1 8 2 3 3 1    <SignatureDefID>
1 5 1 8 2 3 4      <DateTimeStamp>
```

SUMMARY

The pharmaceutical industry in the form of the CDISC consortium has embraced XML as a medium for clinical data. SAS does not provide built-in tools for the importation into, processing of, and exportation of arbitrary, hierarchically structured XML-formatted data. A method is presented for importing, processing, and exporting arbitrary, data-centric XML with SAS. The method uses a canonical document, analogous to an empty record for a row-column dataset, with an indexing algorithm that preserves data hierarchy information to translate XML-formatted data into a flat file representation and vice versa. The flat file representation can be substantially smaller than the XML representation.

CONCLUSION

Regulatory and economic forces in the pharmaceutical industry have begun to motivate the creation of consortium-based clinical data standards that use XML. The technical challenges to bringing XML-formatted data into pre-existing, SAS-based data processing systems are just beginning to be considered and appear to be formidable. Nevertheless, this technical challenge to SAS professionals in the pharmaceutical industry can be successfully met.

REFERENCES

1. Quin, L. Open source XML database toolkit: resources and techniques for improved development. 2000 John Wiley & Sons, Inc., New York, New York.

2. W3C XML pages at www.w3.org/XML

3. Palmer, M. and Hale, C., SAS for statistics; <XML for output>. 2000 PharmaSUG 2000 Proceedings. Zurich Biostatistics, Inc, "Papers and Presentations" www.zbi.net/NewFiles/Presentations.html

4. Tian, F., DeWitt, D. J., Chen, J., and Zhang, C. The design and performance evaluation of alternative XML storage strategies. at www.cs.auc.dk/~tbp/Teaching/DAT7E00/xmlstorage.pdf

5. Bourret, R. XML and Databases. www.rpbourret.com/xml/XMLAndDatabases.htm

6. CDISC Operational Data Model pages at www.cdisc.org/Operational/index.html

ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute Inc.
XML is a trademark of Massachusetts Institute of Technology.
W3C is a registered trademark of the World Wide Web Consortium.
Tekoa Technology is a service mark of Zurich Biostatistics, Inc.
.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the authors at:

Michael Palmer or Cecilia Hale
Zurich Biostatistics, Inc.
45 Park Place South
PMB 178
Morristown, NJ 07960
Phone: 973-727-0025
Email: mcpalmer@zbi.net or cahale@zbi.net
Web: www.zbi.net