Various Techniques for Processing CDISC ODM XML

Chris Olinger, ThotWave Technologies, Cary, NC

ABSTRACT

This paper describes various techniques for processing the CDISC Operational Data Model (ODM) using SAS and Java. The CDISC ODM group's stated mission is to "develop a vendor independent standard model for the interchange and archiving of clinical trials data with study metadata." However, this means little if you cannot get the data out of it and process it effectively using your favorite tools (aka SAS). This paper will deal with the issues and techniques surrounding processing ODM XML for use with SAS.

XML – A BRIEF HISTORY

XML (eXtexsible Markup Language) has its roots in SGML (Standard General Markup Language) – the granduncle of HTML. SGML is a markup language where data is marked up with <tags> used to describe the content and formatting characteristics of the document. When SGML first hit the scene in mid 80's it was touted as *the way* to mark up documents. One of its main strengths was its ability to separate data from presentation, allowing you to format the same document to many output forms. SGML allowed for different rendering mechanisms to consume the same markup document. The end result of the SGML effort was a markup language that was extremely flexible, that tried to be all things to all people, and in the end, horribly complex.

HTML built on the SGML idea by extending it to include the idea of hyper-linking. HTML used a small subset of the available features of SGML, added linking, and boom - the web was born. Unfortunately, not a lot of thought went into the structure and content of HTML. HTML was designed as a rudimentary formatting language that allowed for the connectivity of content in a web browser. There was no separation of data from presentation, no rigorous approach to syntactic definition, and it was caught in the middle of an escalating feature war between Netscape and Microsoft, with each company trying to out-do the other by adding features as fast as they could.

HTML, however, was the fuel of the web revolution. HTML is everywhere now – along with all the problems that go with it. It is very hard, for instance, to remove interesting data from an HTML page because it is intertwined with the formatting commands that the browsers need to render the data. And since HTML is not well formed as a language it is not easily parsed with external tools.

XML is the latest generation of tag-based markup. The implementers of XML envisioned a markup that was well formed, that was human readable, that separated data from display, and that had extensive and easily accessible tools for processing. It took the good things from SGML, the good things from HTML, and mixed them up in an elegant solution. Today, XML is being used for a variety of tasks, but the great majority of those tasks are for data exchange and interoperability. Web visionaries that declared, "The King (HTML) is dead, long live the King (XML)!" are going to be sorely disappointed. Displacing HTML with XML is like displacing gasoline-powered cars with electric cars. It may happen over time, but more than likely the two technologies will exist side by side for a long time.

THE ODM STANDARD

The strong suits of XML are its descriptive nature, its human readability, and its ability to separate data from display. This makes XML attractive for modeling data in an open format. As such, CDISC has chosen XML to define their Operation Data Model (ODM). The ODM provides a definition for study data and study meta-data surrounding a clinical trial. The data contained in the ODM is vendor-system neutral (sometimes in maddening ways), with no preference being given to any one CDMS or processing tool. The model also incorporates extensions for vendor specific data, with the caveat that the XML DTD[1] for the extensions must be supplied and that the extensions can be ignored by systems that do not want to deal with them. It is pre-supposed that not all vendor systems that read ODM XML will be able to process or use all of the information in the model. It is up to the system processing the model to decide what is relevant.

Version 1.0 of ODM XML was released last year. The model itself is fairly complex and at first glance is fairly unintuitive if you are expecting something that reminds you of SAS. CDISC has just released version 1.1, which cleans up the model quite a bit, but it's still hard for someone who lives and breathes SAS. This paper uses the v1.1 model for its examples. You can get all kinds of information on the CDISC model at http://www.cdisc.org/models/odm/v1.1/index.html. The example in the appendix section of this paper is the example posted on the CDISC site.

TIME TO GROK

So, where do we go from here? You have an ODM XML file and you need to get it into SAS. Your boss has given you only a couple days to figure out how to do it, and you only care about some of the data contained in the file (lets say the Demography data). Where to start?

A good starting point is the basic structure of the XML document. If you look at the ODM DTD or an example ODM file you can glean some of the organization:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ODM SYSTEM "odmv11-15-nx.dtd" >

<ODM>
   <Study> … </Study>
   <AdminData> … </AdminData>
   <ReferenceData> … </ReferenceData>
   <ClinicalData> … </ClinicalData>
</ODM>
```

[1] DTD stands for Document Type Definition. The DTD is the definition of what can possibly be included in the XML file. Without it, you are lost in the woods because you have only a limited clue as to what the designers of the XML document had in mind.

The first line of the example describes the file as an XML file. This is standard nomenclature for an XML document. This line must be present. The next line defines the DTD or Document Type Definition file. The DTD is a grammar that defines the allowed elements and attributes in the XML document. In this case, we are using a DTD that describes the ODM XML file without any vendor extensions. Next, we see the <ODM> … </ODM> element tags. The <ODM> element defines the root of the XML document and is required. All of other XML element and attribute tags must fall within the root element. Notice, that all tags in the XML file must have opening tags (<tag>) and closing tags (</tag>). This is part of the requirements for XML. All tags must have a start tag and end tag,[2] or if the tags do not have any content then a shortcut of <tag/> may be used to designate an empty element.

Looking at the above example we see that an ODM XML file consists of zero or more <Study> groupings, zero or more <AdminData> groupings, zero or more <ReferenceData> groupings, and zero or more <ClinicalData> groupings. The <Study> element defines the meta-data that describes the data items contained in the <ClinicalData> elements. <AdminData> defines administrative information about users, locations, and electronic signatures. <ReferenceData> defines information that lets you interpret the clinical data stored in the model such as lab normal ranges.

Simple, right? If this was all there was to an ODM document then it would not be very useful. Most of the work involved in processing an ODM document is in retrieving the data items and meta-data contained in the model. If you look at a simple example (see the Appendix section for an example from CDISC) you'll notice that the data we are interested in is contained in <ClinicalData> section. Specifically, each value point that we are interested in is defined in an <ItemData> element. What is an <ItemData> you ask? An item is an individual clinical item such as a blood pressure reading. <ItemData> is the XML element that models that value.

CDISC has defined a very hierarchical structure for data. As such, an item cannot exist alone in a vacuum - it must be contained in an *item group*. Item groups are grouping of items and can be thought of as an individual record in a table. Each item corresponds to a value in a column. Item groups can be aggregated into *Forms*. A Form can be thought of as a page in a CRF book or screen. A series of forms conspires to be *a Study Event.* A Study Event represents a patient visit or some other collection point, with each event being tagged to a particular patient in the Study. Each study event must be mapped back to the *Study* it came from as well as the *MetaDataVersion* that defined it. To get a single data point for a particular table a processor must traverse a hierarchy that looks something like this:

```
<ClinicalData>
   <SubjectData>
      <StudyEventData>
        <FormData>
          <ItemGroupData>
            <ItemData>
            …
            </ItemData>
          </ItemGroupData>
        </FormData>
      </StudyEventData>
   </SubjectData>
</ClinicalData>
```

This is a lot of work to get a single data point. Also, look at what is actually contained in the item and the item group in our example:

```
<ItemGroupData ItemGroupOID="DEMOG">
   <ItemData ItemOID="PT" Value="P001"/>
</ItemGroupData>
```

The item is tagged with an *ItemOID* attribute that maps the item back to some definition in the study meta-data section. The item group is also mapped back to a meta-data representation. If we look back in the <Study> section of the document then we'll notice elements defining the item group DEMOG and the item PT:

```
<Study …>
<MetaDataVersion …>
…
<!—- Definition of DEMOG item group →
<ItemGroupDef OID="DEMOG" Name="Demographics"
   SASDatasetName="DEMOG">
 <ItemRef ItemOID="PT" Mandatory="No"/>
 <ItemRef ItemOID="INITIALS" Mandatory="No"/>
 <ItemRef ItemOID="SEX" Mandatory="No"/>
 <ItemRef ItemOID="DOB" Mandatory="No"/>
 <ItemRef ItemOID="SPONSOR_PTID" Mandatory="No"/>
 <ItemRef ItemOID="WEIGHT_LB" Mandatory="No"/>
 <ItemRef ItemOID="WEIGHT_KG" Mandatory="No"/>
</ItemGroupDef>
…
<!—- Definition of the PT item →
<ItemDef OID="PT" Name="PT" DataType="text"
  Length="4" SASFieldName="PT"/>
…
</MetaDataVersion>
</Study>
```

The <ItemGroupDef> element defines the meta-data for the DEMOG item group. The definition can be thought of as the configuration of the DEMOG table. The group definition lists a series of <ItemRef> tags that refer to an item defined by an <ItemDef> tag.

At this point, we have enough information to build a SAS data set. We can construct the DEMOG table with the information in the item group definition and item definitions, and we can get the data from the item data elements in the clinical data. There is some other information that we are ignoring (like Code Lists), but we should be able to get a rudimentary table in place. How do we go about it?

[2] Unlike HTML, which allows tags with no closing tag. XHTML attempts to address this by specifying the HTML syntax in a valid XML format.

The ODM models the relationships between clinical data. However, it does not map to a standard SAS representation of data very well at all. SAS data is flat and rectangular, and ODM is very hierarchical in nature. What we need is a way of mapping the hierarchy of ODM to the rectangular structure of SAS. There are many techniques for processing XML. You can use DOM, SAX[3], XSLT, Perl, Java, C++, SAS, Python … the list goes on and on. For our purposes here we are going to concentrate on SAS and Java. XSLT[4] also seems like a natural way to go since its entire existence is predicated on transforming XML, but XSLT is fairly complex and would require an inordinate amount of space to cover in this paper.

XML AND SAS

As of Version 8.2, the production mechanisms for directly reading XML into SAS data sets are limited to an XML libname engine that reads four or five fairly regular XML syntaxes, and reading the XML directly with a data step and input statements. The current libname syntaxes include:

1) Generic – A generic XML format
2) Oracle – An Oracle8iXML flavor
3) OIMDBM – The OIM data model. This model has since been subsumed by OMG
4) Export – An alias for the current flavor of the month. It is currently aliased to OIMDBM
5) HTML – An HTML representation of the data

Unfortunately, these libname types do not come anywhere close to being able to process the ODM model. To process ODM we are going to need another way of getting the data from XML into SAS. For Version 9 of the SAS System (available sometime this summer), the XML libname engine has production support for a set of new functionality called XMLMAP. XMLMAP allows you to describe, via another XML file, how to parse and pull out data elements from an XML document. And luckily, XMLMAP is available for SAS V8.2 in an experimental form. You can download the executables from http://www.sas.com/rnd/base/topics/sxle82/. For documentation on the experimental XMLMAP support you can go to

http://www.sas.com/rnd/base/topics/sxle82/exp82/mapxmlexp.htm

The XMLMAP support is flexible enough to allow us to get ODM XML into SAS datasets. The following is a simple example of how to use the XMLMAP option on the XML libname. For our example, we have a simple XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<body>
   <a name=″Group1″>
     <b name=″B1″> Chris </b>
     <b name=″B2″> Greg </b>
   </a>
   <a name=″Group2″>
     <b name=″B1″> Jeff </b>
     <b name=″B2″> John </b>
   </a>
</body>
```

We would like to turn the above document into a table that contains the following values:

GroupName	RowName	Value
Group1	B1	Chris
Group1	B2	Greg
Group2	B1	Jeff
Group2	B2	John

To do this we need to describe to the XML libname where in the XML document the values live. The description of where the values live is contained in the *map file*. The map file is an XML document that describes which values go where. For the example above, the map file looks like this:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<SXLEMap>
   <TABLE name="Simple">
      <TABLE_XPATH>/body/a/b</TABLE_XPATH>
      <COLUMN name="GroupName" retain="yes">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>15</LENGTH>
         <XPATH>/body/a/@name</XPATH>
      </COLUMN>
      <COLUMN name="RowName" retain="yes">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>15</LENGTH>
         <XPATH>/body/a/b/@name</XPATH>
      </COLUMN>
      <COLUMN name="Value">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>15</LENGTH>
         <XPATH>/body/a/b</XPATH>
      </COLUMN>
   </TABLE>
</SXLEMap>
```

The document starts with “<?xml version="1.0" encoding="iso-8859-1" ?>”. This is standard for all XML documents. The encoding might change but the XML declaration needs to be there. The XML root body element is <SXLEMap>. SXLE stands for “SAS XML Libname Engine”. Next, there is a single definition for a table named “Simple”. The map file supports the definition of many tables, the same way that a libname lets you reference and define multiple data sets. The next element, <TABLE_XPATH> is probably the most import element in the definition. <TABLE_XPATH> tells the libname engine which element or attribute is going to be the output *row iterator*. The value of this

[3] DOM stands for Document Object Model and SAX stands for Simple Api for XML. They are the predominant XML parsing models available today. SAS Institute’s XMLMAP support is based on an underlying SAX parsing model.

[4] XSLT stands for eXtensible Stylesheet Language Transformations. It is a language dedicated to transforming XML to other formats. The language itself is an XML document.

element is “/body/a/b”. What this is saying is that for every XML element in the source document that matches the path “/body/a/b” we are going to execute the column definitions and write a row to the output table. The XPATH component is a path that uniquely addresses the nodes in the XML document. “/body” refers to the <body> root node element, “/body/a” refers to the <a> element contained in the <body> element, and so on.

The remaining <COLUMN> elements describe the output columns that are going to be written to our output table. We define the output column type in <TYPE> as numeric or character. The <DATATYPE> element describes the input type of the value in the input XML document (in this case a string). Our column types are all character so we give a <LENGTH> definition to define the length of the variable in the SAS table. Finally, we point the column to where the actual value is going to be retrieved from every time the <TABLE_XPATH> directive matches a node in the document. You can retrieve values from both elements and attributes. To retrieve a value from an element then you just point to the element name. For instance, “/body/a/b” will retrieve the value that is contained in the <b> node. To retrieve a value from an attribute then you point to the attribute with a @ directive. For instance, “/body/a/b/@name” will retrieve the attribute named “name” from the <b> element.

To use the XML map you just reference it on an XML libname statement. The following code will parse our example document and create our output table:

```
filename SIMPLE   "simple.xml";
filename MAP      "simple.map";
libname  SIMPLE xml xmlmap=MAP;

data simple(label="Simple XML parsed");
  set SIMPLE.Simple;
run;

proc print data=simple; run;
```

The output from this SAS program is listed here:

Obs	GroupName	RowName	Value
1	Group1	B1	Chris
2	Group1	B2	Greg
3	Group1	B1	Jeff
4	Group1	B2	John

We now have a basic understanding of how XMLMAP works. We can apply the same principles here to the data contained in the elements and attributes of the ODM XML model. The plan of attack for this problem is to get all of the data that we think we are going to need from the model into data sets, and then do any remaining patch work with SAS.

If we look at the way the data is structured in the ODM model then we will notice that it’s not possible to lay out a DEMOG or AE data set directly in the XMLMAP file. This restriction is due to the fact that the meta-data about the table is defined in the ODM model itself. This means that the table structure can be different from one ODM model to another. This makes our job a lot tougher. Instead of reading the values of the DEMOG table directly from the XML document, we need to read *all* the values from the ODM and then transpose the values to the correct table in a subsequent step. Our first step, then, is to read the data that we think we are going to need using the XMLMAP syntax.

The following SAS program reads the ODM model and creates various staging tables that we can work with. The XMLMAP file for the ODM model is presented in the Appendix section of this paper.

```
%let home=c:\Thotwave\PharmaSUG;

filename CDISC    "&home\example2.xml";
filename MAP      "&home\CDISC11.map";
libname  CDISC xml xmlmap=MAP;
libname  O        "&home\Output";

data O.GlobalVariables
(label="GlobalVariables");
set CDISC.GlobalVariables; run;
data O.StudyEventRef
(label="StudyEventRef");
set CDISC.StudyEventRef;   run;
data O.StudyEventDef
(label="StudyEventDef");
set CDISC.StudyEventDef;   run;
data O.FormRef
(label="FormRef");
set CDISC.FormRef;         run;
data O.FormDef
(label="FormDef");
set CDISC.FormDef;         run;
data O.ItemGroupRef
(label="ItemGroupRef");
set CDISC.ItemGroupRef;    run;
data O.ItemGroupDef
(label="ItemGroupDef");
set CDISC.ItemGroupDef;    run;
data O.ItemRef
(label="ItemRef");
set CDISC.ItemRef;         run;
data O.ItemDef
(label="ItemDef");
set CDISC.ItemDef;         run;
data O.CodeList
(label="CodeList");
set CDISC.CodeList;        run;
data O.CodeListItem
(label="CodeListItem");
set CDISC.CodeListItem;    run;
data O.SignatureDef
(label="SignatureDef");
set CDISC.SignatureDef;    run;
data O.Location
(label="Location");
set CDISC.Location;        run;
```

```
data O.User
(label="User");
set CDISC.User;            run;
data O.AuditRecord
(label="AutditRecord");
set CDISC.AuditRecord;     run;
data O.Signature
(label="Signature");
set CDISC.Signature;       run;
data O.ItemData
(label="Itemdata");
set CDISC.ItemData;        run;

proc datasets lib=out; run; quit;
```

The above program will create seventeen tables from our input ODM XML file. The tables we are really interested in are the ITEMGROUPDEF, ITEMREF, ITEMDEF, and ITEMDATA tables. The ITEMGROUPDEF table gives us a list of the data tables contained in the model. The ITEMREF table gives us a mapping as to which columns are mapped to which table, and the ITEMDEF table gives us the actual definition of the variables contained in each table. The ITEMDATA table contains the individual data values for each column. Each of these tables references the other tables through the use of various OID fields. The OID fields provide the glue so that you can map values to entries in the item tables.

The ITEMGROUPDEF table looks like this (minus some information not necessary to the example):

ItemGroupOID	SASDatasetName
DEMOG	DEMOG
VITALS	VITALS

The ITEMREF table looks like this (minus some information not necessary to the example):

ItemGroupOID	ItemOID
DEMOG	PT
DEMOG	INITIALS
DEMOG	SEX
DEMOG	DOB
DEMOG	SPONSOR_PTID
DEMOG	WEIGHT_LB
DEMOG	WEIGHT_KG
VITALS	PT
VITALS	VISITNAME
VITALS	SBP
VITALS	DBP
VITALS	SPONSOR_PTID
VITALS	OCCUR_NUM

The ITEMDEF table looks like this (minus some information not necessary to the example):

ItemOID	DataType	SASFieldName
PT	text	PT
INITIALS	text	INITIALS
DOB	date	DOB
SEX	text	SEX
SBP	integer	SBP
DBP	integer	DBP
VISITNAME	text	VISNAME
SPONSOR_PTID	text	SPONPTID
VISITNUM	integer	VISITNUM
OCCUR_NUM	integer	OCCURNUM
WEIGHT_LB	float	WEIGHTLB
WEIGHT_KG	float	WEIGHTKG

A subset (three observations worth) of the ITEMDATA table looks like this (minus some information not necessary to the example):

ItemOID	Value
PT	P001
INITIALS	AMH
SEX	f
DOB	1947-07-16
SPONSOR_PTID	B00-2136-001
WEIGHT_LB	150
WEIGHT_KG	68.18
PT	P001
VISITNAME	Visit1
SBP	120
DBP	80
SPONSOR_PTID	B00-2136-001
OCCUR_NUM	1
PT	P001
VISITNAME	Visit2
SBP	130
DBP	90
SPONSOR_PTID	B00-2136-001
OCCUR_NUM	1

All we need to do now is write some SAS code that reads the meta-data from the SAS data sets and then creates the output in the correct format. This process is left to the reader, although an example is included in the appendix of this paper. The output of the DEMOG table follows:

```
                                The SAS System           11:24 Friday, May 3, 2002   1

                                                                        Weight
                                                 Sponsor      Date of     in      Weight in
Obs     PT     Initials     Sex     Patient ID                Birth    pounds    kilograms

 1     P001      AMH         f     B00-2136-001    16JUL1947     150       68.18
 2     P027      VLP         f     B00-2136-027    20DEC1944     139       63.04
```

While processing the ODM XML model is not trivial with SAS, it is doable. I ran into a number of issues while writing the XMLMAP code that I hope will be cleared up in the V9 production version. The experimental executables in the V8.2 version were fairly stable, but if a problem occurs then SAS goes off and crashes. It is not very forgiving.

Other techniques for processing ODM XML with SAS are to read the XML file directly and store values for each node in the hierarchy in a collection of arrays. If you also maintain the parent child relationships then you can process the file and also have the information available to write the ODM XML back out. Unfortunately, this method requires you to write a lot of parsing code, which is not very nice. As an aside, the technique that we describe here is fairly destructive. If you want to write the XML document back out then you need to be very careful (more careful than I was) at parsing all of the information in the model and storing it in a table. A round trip is a harder problem than just pulling out needed information.

XML AND JAVA

You may have evaluated the technique above and decided that it does not work for you. You may still be running V6.12 of the SAS System[5] or you do not want to use an experimental set of software until V9.0 is released and validated. Either way, if you don't want to use SAS then you need to turn to other technologies. I have heard it said that SAS programmers do not like to venture out of SAS – and this may be true. However, my philosophy is that you use the tool that suits your needs best. A programming language is just like a hammer. Sometimes you need a different type of hammer to build your house.

As I mentioned before, there are various technologies for parsing and processing XML. The three main technologies are DOM, SAX, and XSLT. The DOM technique reads the entire XML tree into a *node-set* in memory. You can manipulate this node-set with method calls. Various languages support a DOM API. The DOM tends to be unwieldy and a memory hog. The SAX technique provides a callback mechanism. You register which node types you are interested in and when the parser encounters them it calls your routines. This is a great way to go if you can process the entire file in one pass. It's fast, it does not use a lot of memory, and there is a multitude of support out there from all the major languages. However, if you need to do multiple passes or retain state then you may have to do a lot extra work. XSLT was made precisely for transforming XML, but it tends to be slow. Also, XSLT syntax takes quite a bit of getting used to, as it is a pattern-matching declarative language instead of a procedural language.

My preference is to use Java. It is fast, reliable, readily available, and has some cool technology built in already that make our life easier. My preference also is to *not* use DOM or SAX because I do not want to write a lot of messy parsing code. Instead of DOM or SAX, I like to use something called JAXB.[6] JAXB may be one of the coolest technologies to happen to XML in a long time.

JAXB allows you to treat your XML document as series of Java objects. You tell JAXB the DTD that it should use to build the XML binding classes, and then the JAXB compiler generates code for you that automatically parses the XML file and creates a set of objects. This technique allows you to stay away from the XML and just call methods to get the information that you need. JAXB is a little like DOM and SAX all rolled into one. It stores information in memory like a DOM, but it does not need to store as much information as a normal DOM. It parses like SAX but it allows you to go back and forth across the tree as many times as you like. Finally, you can configure JAXB in a multitude of ways to generate precisely the code that you want it to generate. For this paper, I have stuck to the basic options. You can find more information about JAXB at the Sun website:
http://java.sun.com/xml/jaxb/

Our goal then is to use JAXB to parse and process our ODM XML document. Parsing and reading the file are easy. How do we then get the data into a SAS data set? The technique used here is to use JAXB to write a series of data step code files. Each unique data set referenced in the ODM document will generate its own external file. One of the nice things about this approach is that you have an independent SAS program that you can validate, as well as a precise record of what the program generated.

The first step is to tell the JAXB compiler what kind of classes to generate. We do this with a configuration XML file:

```
<xml-java-binding-schema version="1.0ea">
   <options
      package='com.thotwave.odm'
      default-reference-collection-type='array'
      property-get-set-prefixes='true'
      marshallable='false'
      unmarshallable='true'
      />
   <element name="ODM" type="class" root="true"/>
</xml-java-binding-schema>
```

[5] Why?

[6] JAXB - Java Architecture for XML Binding

The above XML file specifies that the generated code use the "com.thotwave.odm" package name, that lists of objects should use the Java array type, that set and get methods should be generated for all attributes, and whether we will be reading or writing XML from the classes. The final entry in the file is a directive telling JAXB that the main root object in the file is the <ODM> XML element. You can point JAXB to multiple roots in a file if you would like to do so. As mentioned previously, this configuration is pretty basic. You can also tell JAXB the class names to use for various elements in the document.

You feed this configuration file to the JAXB compiler along with the DTD of the XML file that you want to parse. The compiler responds by generating a bunch of classes for you to use.

```
xjc odmv11-15-nx.dtd odm.xjs

com\thotwave\odm\Address.java
com\thotwave\odm\AdminData.java
com\thotwave\odm\Annotation.java
com\thotwave\odm\ArchiveLayout.java
com\thotwave\odm\ArchiveLayoutRef.java
com\thotwave\odm\AuditRecord.java
com\thotwave\odm\BasicDefinitions.java
com\thotwave\odm\ClinicalData.java
com\thotwave\odm\CodeList.java
com\thotwave\odm\CodeListItem.java
com\thotwave\odm\CodeListRef.java
com\thotwave\odm\Comment.java
com\thotwave\odm\Decode.java
com\thotwave\odm\ErrorMessage.java
com\thotwave\odm\ExternalCodeList.java
com\thotwave\odm\ExternalQuestion.java
com\thotwave\odm\Flag.java
com\thotwave\odm\FlagType.java
com\thotwave\odm\FlagValue.java
com\thotwave\odm\FormData.java
com\thotwave\odm\FormDef.java
…
```

In all, fifty-six classes are generated. Now all we have to do is compile the classes and write the code that will access the data stored in these classes. The following section of code shows how to instantiate the root node <ODM> class:

```
/**
 * Build a SAS program from an ODM file
 */
public static void main( String[] argv ) {

   // get the file
   if ( argv.length == 0 ) {
       System.out.println( "No file to parse!" );
       return;
   }

   // load the application.xml file
   ODM odm;
   try {
       InputStream is = new
           FileInputStream(argv[0]);

       // read the xml file
       // Unmarshall the march.xml file
       odm = new ODM();
       odm = odm.unmarshal(is);

       // close the stream
       is.close();

       // ready to roll
       System.out.println(
         "Parsed successfully!" );
       System.out.println(
         "Writing: " + argv[1] );

       // write the SAS job
       writeSAS( odm, argv[1] );
   }
   catch( Exception e ) {
      e.printStackTrace();
      return;
   }
}
```

The ODM class that is instantiated is generated as part of the JAXB compilation step. There is one class generated for every distinct element that exists in the ODM model. The root that you choose in the JAXB configuration file is the class that you call *unmarshal()* on. The unmarshal method takes an input stream as input and expects that the stream points to an ODM XML document. If there is a problem parsing the document then the code throws an exception which is handled by the *try {} catch ()* block. If all goes well then we have the ODM document in memory and we can now call methods on the ODM class to retrieve the values contained in the model. The *writeSAS()* method just reads through the object tree and builds a set of data step code that builds the data sets we are interested in. Here is a snippet from the writeSAS() method:

```
/**
 * Write the odm file
 */
static private void writeSAS( ODM odm,
                              String outfile )
   throws IOException
{
   // This map tracks all of our files
   Map map = new HashMap();

   // process the metadata
   processStudies( map, odm );
```

```
   // process the data
   processClinicalData( map, odm );

   // move the files from temp space to permanent
   createFiles( map, outfile );
}

// process the study list
static private void processStudies( Map map,
                                    ODM odm )
   throws IOException
{
   // do it across all studies
   Study[] studyList = odm.getStudy();
   if ( studyList == null ) return;

   for ( int i = 0; i < studyList.length; i++ )
       processMetaDataVersion( map, odm,
                               studyList[i] );
}

// process the clinical data
static private void processClinicalData(
                       Map map, ODM odm )
{
   ClinicalData[] cdList = odm.getClinicalData();
   if ( cdList == null ) return;

   // loop across clinical data
   for ( int i = 0; i < cdList.length; i++ )
       processSubjectData( map, odm, cdList[i] );
}
```

The methods are simple. We process the meta-data first in *processStudies(map, odm)* to get our definitions and then we retrieve the values for the data in the *processClinicalData(map, odm).* After we are done, we write our temp files to permanent files. *processStudies()* just loops through the study list that is contained in our ODM model (zero or more studies may be present). For each study in the model we call *processMetaDataVersion().* Also, notice how the clinical data values tend to mimic the structure of the meta-data. The *processClinicalData()* retrieves the clinical data nodes in the ODM file and then processes them. This recursive process is repeated for all of the nodes that we are interested in the ODM document. As we descend the tree of nodes we collect data from the meta-data and build data step code with the values from the <ClinicalData> section. The end result is a SAS file that looks like this:

```
data DEMOG(
   label="Study(123-456-789)
          MetaDataVersion(MetaDataVersion.001)
          Demographics");
   length PT $4;
   attrib PT label="PT";
   length INITIALS $5;
   attrib INITIALS label="Initials";
   length SEX $1;
   attrib SEX label="Sex";
   attrib DOB format=YYMMDD10.;
   attrib DOB label="Date of Birth";
   length SPONPTID $12;
   attrib SPONPTID label="Sponsor Patient ID";
   attrib WEIGHTLB label="Weight in pounds";
   attrib WEIGHTKG label="Weight in kilograms";
   PT='P001';
   INITIALS='AMH';
   SEX='f';
   DOB=input('1947-07-16',YYMMDD10.);
   SPONPTID='B00-2136-001';
   WEIGHTLB=150;
   WEIGHTKG=68.18;
   output;
   PT='P027';
   INITIALS='VLP';
   SEX='f';
   DOB=input('1944-12-20',YYMMDD10.);
   SPONPTID='B00-2136-027';
   WEIGHTLB=139;
   WEIGHTKG=63.04;
   output;
run;
```

The code that produced the above output totaled around 350 lines of code.

CONCLUSION

There are various methods for processing CDISC ODM XML. The nature of the model requires that you use clever techniques for reading the data. SAS can be used, but it is currently difficult to read data that is unabashedly hierarchical. V9 of the SAS System and the new SXLE XMLMAP option promises to simplify this process a great deal, but you still have to write transformation code.

Alternative methodologies can also be used to process the model. Java is an industry standard that matches well with XML and can be used to transform the data into any format that you like.

ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute Inc.

Java and JAXB are registered trademarks of Sun Microsystems.

The SAS code for transforming the XMLMAP output set was written by Jack Shoemaker (SAS hacker extraordinaire). Jack you are the best!

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Please feel free to contact the author at:

Chris Olinger
colinger@thotwave.com
117 Edinburgh South
Suite 202
Cary, NC 27511
919.465.0322 - Phone
919.465.0323 - Fax

APPENDIX

EXAMPLE 1: A SAMPLE FROM CDISC

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ODM SYSTEM "odmv11-15-nx.dtd" >
<!--
     This XML Example is intended to provide a very small, valid example of a CDISC study
     using CDISC ODM V1.1
-->
<!--
     In this small example study, a pharmaceutical company is conducting a Phase II initial
     efficacy study.  This pharmaceutical company ( "The Pharm" ) has hired a contract
     research organization (CRO) to collect, clean and complete data for this study.  Pharm
     and the CRO use completely different clinical data management systems, so the 2 companies
     have agreed to use CDISC'S ODM V1.1 for their data exchange.

     The data are coming in slowly and only the CRF pages from the first two visits for two
     patients have been returned from the investigational site to the CRO so far.  During the
     first visit, both demographic and vital statistics data are collected. During the second
     visit, only  vital statistics are collected.  The data from the first patient's first visit
     included an incorrect systolic blood pressure. The incorrect value was changed to the
     correct value in a separate data entry session, and this change was captured in the audit
     trail of the clinical data management system.
-->
<ODM FileType="Transactional" FileOID="987-654-321" CreationDateTime="2001-11-15T16:37:21-05:00"
     AsOfDateTime="2001-11-15T16:37:21-05:00"
     Description="Two visits for patients P001 and P027, plus updates of data for P001"
     Granularity="All">
  <Study OID="123-456-789">
    <GlobalVariables>
      <StudyName>HarrisonA</StudyName>
      <StudyDescription>Phase II initial efficacy</StudyDescription>
      <ProtocolName>B00-2136</ProtocolName>
    </GlobalVariables>
    <BasicDefinitions/>
    <!-- Beginning of metadata definitions . . . -->
    <MetaDataVersion OID="MetaDataVersion.001" Name="Release 1">
      <Protocol>
        <!-- There are 2 study events defined, corresponding to Visit 1 and Visit 2.
             On "Demog and Vitials":, signing unit is used, while on Visit 2, the "Vitals Only"
             is used
        -->
        <StudyEventRef StudyEventOID="VISIT_1" Mandatory="Yes"/>
        <StudyEventRef StudyEventOID="VISIT_2" Mandatory="Yes"/>
      </Protocol>
      <StudyEventDef OID="VISIT_1" Name="Visit1" Repeating="No" Type="Scheduled">
```

```
  <FormRef FormOID="PAGE_1" Mandatory="No"/>
</StudyEventDef>
<StudyEventDef OID="VISIT_2" Name="Visit2" Repeating="No" Type="Scheduled">
  <FormRef FormOID="PAGE_2" Mandatory="No"/>
</StudyEventDef>
<!-- There are 2 signing units defined: one to collect demograhics AND vitals and
     one to collect just vitals.
  -->
<FormDef OID="PAGE_1" Name="Demog and Vitals" Repeating="No">
  <ItemGroupRef ItemGroupOID="DEMOG" Mandatory="No"/>
  <ItemGroupRef ItemGroupOID="VITALS" Mandatory="No"/>
</FormDef>
<FormDef OID="PAGE_2" Name="Vitals Only" Repeating="No">
  <ItemGroupRef ItemGroupOID="VITALS" Mandatory="No"/>
</FormDef>
<!--There are 2 item groups (ItemGroups): Demographics and Vitals -->
<!--The demographics item group contains 7 item references (ItemRef) corresponding to
    Patient ID, Patient Initials, Sex, Date of Birth, Sponsor Patient ID, Weight (lb)
    and Weight (kg)-->
<ItemGroupDef OID="DEMOG" Name="Demographics" Repeating="Yes" IsReferenceData="No"
  SASDatasetName="DEMOG">
  <ItemRef ItemOID="PT" Mandatory="No"/>
  <ItemRef ItemOID="INITIALS" Mandatory="No"/>
  <ItemRef ItemOID="SEX" Mandatory="No"/>
  <ItemRef ItemOID="DOB" Mandatory="No"/>
  <ItemRef ItemOID="SPONSOR_PTID" Mandatory="No"/>
  <ItemRef ItemOID="WEIGHT_LB" Mandatory="No"/>
  <ItemRef ItemOID="WEIGHT_KG" Mandatory="No"/>
</ItemGroupDef>
<!-- The Vitals item group contains 6 item references (ItemRef) corresponding to the
     Patient ID, Visit Name, Systolic Blood Pressure, Diastolic Blood Pressure,
     Sponsor Patient ID, and Occurrence Number (of repeat measurement).-->
<ItemGroupDef OID="VITALS" Name="Vital Signs" Repeating="Yes" IsReferenceData="No"
  SASDatasetName="VITALS">
  <ItemRef ItemOID="PT" Mandatory="No"/>
  <ItemRef ItemOID="VISITNAME" Mandatory="No"/>
  <ItemRef ItemOID="SBP" Mandatory="No"/>
  <ItemRef ItemOID="DBP" Mandatory="No"/>
  <ItemRef ItemOID="SPONSOR_PTID" Mandatory="No"/>
  <ItemRef ItemOID="OCCUR_NUM" Mandatory="No"/>
</ItemGroupDef>
<!--There are 12 items defined: Patient ID, Initials, Date of Birth, Sex, Systolic
    Blood Pressure, Diastolic Blood Pressure, Visit Name, Sponsor Patient ID, Visit
    Number (not currently used), Occurrence NUmber, Weight (lb) and Weight (kg).-->
<ItemDef OID="PT" Name="PT" DataType="text" Length="4" SASFieldName="PT"/>
<ItemDef OID="INITIALS" Name="Initials" DataType="text" Length="5" SASFieldName="INITIALS"/>
<ItemDef OID="DOB" Name="Date of Birth" DataType="date" SASFieldName="DOB"/>
<ItemDef OID="SEX" Name="Sex" DataType="text" Length="1" SASFieldName="SEX">
  <CodeListRef CodeListOID="CodeList.001"/>
</ItemDef>
<ItemDef OID="SBP" Name="Systolic Blood Pressure" DataType="integer"
   SASFieldName="SBP"/>
<ItemDef OID="DBP" Name="Diastolic Blood Pressure" DataType="integer"
   SASFieldName="DBP"/>
<ItemDef OID="VISITNAME" Name="Visit" DataType="text" Length="20"
```

```
      SASFieldName="VISNAME"/>
    <ItemDef OID="SPONSOR_PTID" Name="Sponsor Patient ID" DataType="text" Length="12"
      SASFieldName="SPONPTID"/>
    <ItemDef OID="VISITNUM" Name="Numeric Version fo Visit" DataType="integer"
      SASFieldName="VISITNUM"/>
    <ItemDef OID="OCCUR_NUM" Name="Occurence Number" DataType="integer"
      SASFieldName="OCCURNUM"/>
    <ItemDef OID="WEIGHT_LB" Name="Weight in pounds" DataType="float"
      SASFieldName="WEIGHTLB"/>
    <ItemDef OID="WEIGHT_KG" Name="Weight in kilograms " DataType="float"
      SASFieldName="WEIGHTKG"/>
    <!--There is a single codelist for Sex with entries for "m" and "f".-->
    <CodeList OID="CodeList.001" Name="Sex" DataType="text" SASFormatName="CL001">
      <CodeListItem CodedValue="m">
        <Decode>
          <TranslatedText xml:lang="en">Male</TranslatedText>
        </Decode>
      </CodeListItem>
      <CodeListItem CodedValue="f">
        <Decode>
          <TranslatedText xml:lang="en">Female</TranslatedText>
        </Decode>
      </CodeListItem>
    </CodeList>
    <!--Since structure and content of Presentation have not yet been defined,
      we have inserted an empty Presentation element.-->
    <Presentation OID=""/>
  </MetaDataVersion>
</Study>
<!--Beginning of Administrative Data-->
<AdminData>
  <!--There are 2 users: John Doe (the investigator) and Mary Brown (the data entry person)-->
  <User OID="User.001">
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
  </User>
  <User OID="User.002">
    <FirstName>Mary</FirstName>
    <LastName>Brown</LastName>
  </User>
  <!--There is 1 location: General Hospital (a site).-->
  <Location OID="Location.001" Name="General Hospital" LocationType="Site">
    <MetaDataVersionRef StudyOID="123-456-789" MetaDataVersionOID="MetaDataVersion.001"
      EffectiveDate="2001-10-19T10:45:57-05:00"/>
  </Location>
  <!--There is one signature definition (electronic signature for data entry) -->
  <SignatureDef OID="SignatureDef.001" Methodology="Electronic">
    <Meaning>Entry</Meaning>
    <LegalReason>Signer accepts responsibility for initial data entry</LegalReason>
  </SignatureDef>
</AdminData>
<!--Beginning of Clinical Data-->
<ClinicalData StudyOID="123-456-789" MetaDataVersionOID="MetaDataVersion.001">
  <!-- SubjectData element for the first of 2 patients in this example  -->
  <SubjectData SubjectKey="P001" TransactionType="Insert">
```

```
<!--  EventData element for Visit 1 for this patient  -->
<StudyEventData StudyEventOID="VISIT_1">
  <!-- FormData element for the DEMOG and Vitals form collected on Visit 1 (Original)  -->
  <FormData FormOID="PAGE_1">
    <AuditRecord>
      <UserRef UserOID="User.002"/>
      <LocationRef LocationOID="Location.001"/>
      <DateTimeStamp>2001-05-31-T10:08:40-05:00</DateTimeStamp>
    </AuditRecord>
    <!--  Electronic signature for this form   -->
    <Signature>
      <UserRef UserOID="User.001"/>
      <LocationRef LocationOID="Location.001"/>
      <SignatureRef SignatureOID="SignatureDef.001"/>
      <DateTimeStamp>2001-05-30T10:06:32-05:00</DateTimeStamp>
    </Signature>
    <!--  ItemGroupData element for the Demographics part of the form   -->
    <ItemGroupData ItemGroupOID="DEMOG" ItemGroupRepeatKey="1">
      <ItemData ItemOID="PT" Value="P001"/>
      <ItemData ItemOID="INITIALS" Value="AMH"/>
      <ItemData ItemOID="SEX" Value="f"/>
      <ItemData ItemOID="DOB" Value="1947-07-16"/>
      <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-001"/>
      <ItemData ItemOID="WEIGHT_LB" Value="150"/>
      <ItemData ItemOID="WEIGHT_KG" Value="68.18"/>
      <!--  ItemGroupData element for the Vitals part of the form     -->
    </ItemGroupData>
    <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="">
      <ItemData ItemOID="PT" Value="P001"/>
      <ItemData ItemOID="VISITNAME" Value="Visit1"/>
      <ItemData ItemOID="SBP" Value="120"/>
      <ItemData ItemOID="DBP" Value="80"/>
      <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-001"/>
      <ItemData ItemOID="OCCUR_NUM" Value="1"/>
    </ItemGroupData>
  </FormData>
  <!--  FormData element for the Vitals form collected on Visit 1 (Modified)    -->
</StudyEventData>
<!-- StudyEventData element for Visit 2 for the patient  -->
<StudyEventData StudyEventOID="VISIT_2">
  <!--  FormData element for the Vitals only form collected on Visit 2  -->
  <FormData FormOID="PAGE_2">
    <AuditRecord>
      <UserRef UserOID="User.002"/>
      <LocationRef LocationOID="Location.001"/>
      <DateTimeStamp>2001-05-31T9:06:45-05:00</DateTimeStamp>
    </AuditRecord>
    <!-- Electronic signature for this form -->
    <Signature>
      <UserRef UserOID="User.001"/>
      <LocationRef LocationOID="Location.001"/>
      <SignatureRef SignatureOID="SignatureDef.001"/>
      <DateTimeStamp>2001-05-30T10:06:32-05:00</DateTimeStamp>
    </Signature>
    <!--  Data for 3 repeats of Vitals item group on this form -->
```

```
      <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="1">
        <ItemData ItemOID="PT" Value="P001"/>
        <ItemData ItemOID="VISITNAME" Value="Visit2"/>
        <ItemData ItemOID="SBP" Value="130"/>
        <ItemData ItemOID="DBP" Value="90"/>
        <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-001"/>
        <ItemData ItemOID="OCCUR_NUM" Value="1"/>
      </ItemGroupData>
      <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="2">
        <ItemData ItemOID="PT" Value="P001"/>
        <ItemData ItemOID="VISITNAME" Value="Visit2"/>
        <ItemData ItemOID="SBP" Value="125"/>
        <ItemData ItemOID="DBP" Value="86"/>
        <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-001"/>
        <ItemData ItemOID="OCCUR_NUM" Value="2"/>
      </ItemGroupData>
      <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="3">
        <ItemData ItemOID="PT" Value="P001"/>
        <ItemData ItemOID="VISITNAME" Value="Visit2"/>
        <ItemData ItemOID="SBP" Value="122"/>
        <ItemData ItemOID="DBP" Value="84"/>
        <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-001"/>
        <ItemData ItemOID="OCCUR_NUM" Value="3"/>
      </ItemGroupData>
    </FormData>
  </StudyEventData>
</SubjectData>
<SubjectData SubjectKey="P001" TransactionType="Update">
  <StudyEventData StudyEventOID="VISIT_1">
    <FormData FormOID="Page 1">
      <AuditRecord>
        <UserRef UserOID="User.002"/>
        <LocationRef LocationOID="Location.001"/>
        <DateTimeStamp>2001-05-31T08:47:07-05:00</DateTimeStamp>
        <ReasonForChange>Error in initial data entry</ReasonForChange>
      </AuditRecord>
      <Signature>
        <UserRef UserOID="User.001"/>
        <LocationRef LocationOID="Location.001"/>
        <SignatureRef SignatureOID="SignatureDef.001"/>
        <DateTimeStamp>2001-05-30T10:06:32-05:00</DateTimeStamp>
      </Signature>
      <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="1">
        <ItemData ItemOID="SBP" Value="130"/>
      </ItemGroupData>
    </FormData>
  </StudyEventData>
</SubjectData>
<SubjectData SubjectKey="P027" TransactionType="Insert">
  <!--  EventData element for Visit 1 for this patient  -->
  <StudyEventData StudyEventOID="VISIT_1">
    <!-- FormData element for the DEMOG and Vitals form collected on Visit 1 (Original)  -->
    <FormData FormOID="PAGE_1">
      <AuditRecord>
        <UserRef UserOID="User.002"/>
```

```
      <LocationRef LocationOID="Location.001"/>
      <DateTimeStamp>2001-05-31-T10:08:40-05:00</DateTimeStamp>
    </AuditRecord>
    <!--  Electronic signature for this form   -->
    <Signature>
      <UserRef UserOID="User.001"/>
      <LocationRef LocationOID="Location.001"/>
      <SignatureRef SignatureOID="SignatureDef.001"/>
      <DateTimeStamp>2001-05-30T10:06:32-05:00</DateTimeStamp>
    </Signature>
    <!--  ItemGroupData element for the Demographics part of the form   -->
    <ItemGroupData ItemGroupOID="DEMOG" ItemGroupRepeatKey="1">
      <ItemData ItemOID="PT" Value="P027"/>
      <ItemData ItemOID="INITIALS" Value="VLP"/>
      <ItemData ItemOID="SEX" Value="f"/>
      <ItemData ItemOID="DOB" Value="1944-12-20"/>
      <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-027"/>
      <ItemData ItemOID="WEIGHT_LB" Value="139"/>
      <ItemData ItemOID="WEIGHT_KG" Value="63.04"/>
      <!--  ItemGroupData element for the Vitals part of the form     -->
    </ItemGroupData>
    <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="">
      <ItemData ItemOID="PT" Value="P027"/>
      <ItemData ItemOID="VISITNAME" Value="Visit1"/>
      <ItemData ItemOID="SBP" Value="115"/>
      <ItemData ItemOID="DBP" Value="75"/>
      <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-027"/>
      <ItemData ItemOID="OCCUR_NUM" Value="1"/>
    </ItemGroupData>
  </FormData>
  <!--  FormData element for the Vitals form collected on Visit 1 (Modified)    -->
</StudyEventData>
<!-- StudyEventData element for Visit 2 for the patient  -->
<StudyEventData StudyEventOID="VISIT_2">
  <!--  FormData element for the Vitals only form collected on Visit 2  -->
  <FormData FormOID="PAGE_2">
    <AuditRecord>
      <UserRef UserOID="User.002"/>
      <LocationRef LocationOID="Location.001"/>
      <DateTimeStamp>2001-05-31T9:06:45-05:00</DateTimeStamp>
    </AuditRecord>
    <!-- Electronic signature for this form -->
    <Signature>
      <UserRef UserOID="User.001"/>
      <LocationRef LocationOID="Location.001"/>
      <SignatureRef SignatureOID="SignatureDef.001"/>
      <DateTimeStamp>2001-05-30T10:06:32-05:00</DateTimeStamp>
    </Signature>
    <!--  Data for 3 repeats of Vitals item group on this form -->
    <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="1">
      <ItemData ItemOID="PT" Value="P027"/>
      <ItemData ItemOID="VISITNAME" Value="Visit2"/>
      <ItemData ItemOID="SBP" Value="120"/>
      <ItemData ItemOID="DBP" Value="90"/>
      <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-027"/>
```

```
            <ItemData ItemOID="OCCUR_NUM" Value="1"/>
          </ItemGroupData>
          <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="2">
            <ItemData ItemOID="PT" Value="P027"/>
            <ItemData ItemOID="VISITNAME" Value="Visit2"/>
            <ItemData ItemOID="SBP" Value="120"/>
            <ItemData ItemOID="DBP" Value="80"/>
            <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-027"/>
            <ItemData ItemOID="OCCUR_NUM" Value="2"/>
          </ItemGroupData>
          <ItemGroupData ItemGroupOID="VITALS" ItemGroupRepeatKey="3">
            <ItemData ItemOID="PT" Value="P027"/>
            <ItemData ItemOID="VISITNAME" Value="Visit2"/>
            <ItemData ItemOID="SBP" Value="125"/>
            <ItemData ItemOID="DBP" Value="80"/>
            <ItemData ItemOID="SPONSOR_PTID" Value="B00-2136-027"/>
            <ItemData ItemOID="OCCUR_NUM" Value="3"/>
          </ItemGroupData>
        </FormData>
      </StudyEventData>
    </SubjectData>
  </ClinicalData>
</ODM>
```

EXAMPLE 2: THE XML MAP TO READ THE SAMPLE

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<SXLEMap>

   <!--  +=========================================================+ -->
   <!--  + Global Variables                                        + -->
   <!--  +=========================================================+ -->
   <TABLE name="GlobalVariables">
   <TABLE_DESCRIPTION>GlobalVariables</TABLE_DESCRIPTION>
      <TABLE_XPATH>/ODM/Study/GlobalVariables</TABLE_XPATH>
      <TABLE_END_XPATH BeginEnd="Begin">/ODM/Study/BasicDefinitions</TABLE_END_XPATH>

      <COLUMN name="StudyOID" retain="yes">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>20</LENGTH>
         <XPATH>/ODM/Study/@OID</XPATH>
      </COLUMN>

      <COLUMN name="StudyName">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>40</LENGTH>
         <XPATH>/ODM/Study/GlobalVariables/StudyName</XPATH>
      </COLUMN>

      <COLUMN name="StudyDescription">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>200</LENGTH>
```

```
      <XPATH>/ODM/Study/GlobalVariables/StudyDescription</XPATH>
   </COLUMN>

   <COLUMN name="ProtocolName">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/GlobalVariables/ProtocolName</XPATH>
   </COLUMN>
</TABLE>

<!--  +=========================================================+ -->
<!--  + Study Event Ref                                         + -->
<!--  +=========================================================+ -->
<TABLE name="StudyEventRef">
<TABLE_DESCRIPTION>StudyEventRef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/Protocol/StudyEventRef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="StudyEventOID">
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/Protocol/StudyEventRef/@StudyEventOID</XPATH>
   </COLUMN>
</TABLE>

<!--  +=========================================================+ -->
<!--  + StudyEventDef                                           + -->
<!--  +=========================================================+ -->
<TABLE name="StudyEventDef">
<TABLE_DESCRIPTION>StudyEventDef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/StudyEventDef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
```

```
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="StudyEventOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Name">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/@Name</XPATH>
   </COLUMN>

   <COLUMN name="Repeating">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/@Repeating</XPATH>
   </COLUMN>

   <COLUMN name="Type">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/@Type</XPATH>
   </COLUMN>
</TABLE>

<!--  +=============================================================+ -->
<!--  + FormRef                                                     + -->
<!--  +=============================================================+ -->
<TABLE name="FormRef">
<TABLE_DESCRIPTION>FormRef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/FormRef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>
```

```
   <COLUMN name="StudyEventOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="FormOID">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/StudyEventDef/FormRef/@FormOID</XPATH>
   </COLUMN>
</TABLE>

<!--  +=========================================================+ -->
<!--  + FormDef                                                 + -->
<!--  +=========================================================+ -->
<TABLE name="FormDef">
<TABLE_DESCRIPTION>FormDef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/FormDef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="FormOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/FormDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Name">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/FormDef/@Name</XPATH>
   </COLUMN>

   <COLUMN name="Repeating">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
```

```
      <XPATH>/ODM/Study/MetaDataVersion/FormDef/@Repeating</XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + ItemGroupRef                                            + -->
<!--  +=========================================================+ -->
<TABLE name="ItemGroupRef">
<TABLE_DESCRIPTION>ItemGroupRef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/FormDef/ItemGroupRef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="FormOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/FormDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="ItemGroupOID">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/FormDef/ItemGroupRef/@ItemGroupOID</XPATH>
   </COLUMN>
</TABLE>

<!--  +=========================================================+ -->
<!--  + ItemGroupDef                                            + -->
<!--  +=========================================================+ -->
<TABLE name="ItemGroupDef">
<TABLE_DESCRIPTION>ItemGroupDef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>
```

```
   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="ItemGroupOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Name">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@Name</XPATH>
   </COLUMN>

   <COLUMN name="Repeating">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@Repeating</XPATH>
   </COLUMN>

   <COLUMN name="IsReferenceData">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@IsReferenceData</XPATH>
   </COLUMN>

   <COLUMN name="SASDatasetName">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@SASDatasetName</XPATH>
   </COLUMN>

</TABLE>

<!--  +=============================================================+ -->
<!--  + ItemRef                                                     + -->
<!--  +=============================================================+ -->
<TABLE name="ItemRef">
<TABLE_DESCRIPTION>ItemRef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/ItemRef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
```

```
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="ItemGroupOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="ItemOID">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemGroupDef/ItemRef/@ItemOID</XPATH>
   </COLUMN>
</TABLE>

<!--  +=============================================================+ -->
<!--  + ItemDef                                                     + -->
<!--  +=============================================================+ -->
<TABLE name="ItemDef">
<TABLE_DESCRIPTION>ItemDef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/ItemDef</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="ItemOID" retain="yes">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Name">
      <type> character </type>
```

```
      <DATATYPE>string</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemDef/@Name</XPATH>
   </COLUMN>

   <COLUMN name="DataType">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemDef/@DataType</XPATH>
   </COLUMN>

   <COLUMN name="SASFieldName">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemDef/@SASFieldName</XPATH>
   </COLUMN>

   <COLUMN name="CodeListOID">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/ItemDef/CodeListRef/@CodeListOID</XPATH>
   </COLUMN>

</TABLE>

<!--  +=============================================================+ -->
<!--  + CodedList                                                   + -->
<!--  +=============================================================+ -->
<TABLE name="CodeList">
<TABLE_DESCRIPTION>CodeList</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/Study/MetaDataVersion/CodeList</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/@OID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
   </COLUMN>

   <COLUMN name="CodeListOID">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/CodeList/@OID</XPATH>
   </COLUMN>
```

```
   <COLUMN name="Name">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/CodeList/@Name</XPATH>
   </COLUMN>

   <COLUMN name="DataType">
      <type> character </type>
      <DATATYPE>string</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/Study/MetaDataVersion/CodeList/@DataType</XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + User                                                    + -->
<!--  +=========================================================+ -->
<TABLE name="User">
<TABLE_DESCRIPTION>User</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/AdminData/User</TABLE_XPATH>

   <COLUMN name="UserOID">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/AdminData/User/@OID</XPATH>
   </COLUMN>

   <COLUMN name="FirstName">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/AdminData/User/FirstName</XPATH>
   </COLUMN>

   <COLUMN name="LastName">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/AdminData/User/LastName</XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + Location                                                + -->
<!--  +=========================================================+ -->
<TABLE name="Location">
<TABLE_DESCRIPTION>Location</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/AdminData/Location</TABLE_XPATH>

   <COLUMN name="LocationOID">
```

```
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/AdminData/Location/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Name">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/AdminData/Location/@Name</XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + SignatureDef                                            + -->
<!--  +=========================================================+ -->
<TABLE name="SignatureDef">
<TABLE_DESCRIPTION>SignatureDef</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/AdminData/SignatureDef</TABLE_XPATH>

   <COLUMN name="SignatureOID">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/AdminData/SignatureDef/@OID</XPATH>
   </COLUMN>

   <COLUMN name="Meaning">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/AdminData/SignatureDef/Meaning</XPATH>
   </COLUMN>

   <COLUMN name="Legal">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>80</LENGTH>
      <XPATH>/ODM/AdminData/SignatureDef/LegalReason</XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + AuditRecord                                             + -->
<!--  +=========================================================+ -->
<TABLE name="AuditRecord">
<TABLE_DESCRIPTION>FormInfo</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
```

```
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/@StudyOID</XPATH>
</COLUMN>

<COLUMN name="MetaDataVersionOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/@MetaDataVersionOID</XPATH>
</COLUMN>

<COLUMN name="SubjectKey" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/@SubjectKey</XPATH>
</COLUMN>

<COLUMN name="TransactionType" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/@TransactionType</XPATH>
</COLUMN>

<COLUMN name="StudyEventOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/@StudyEventOID</XPATH>
</COLUMN>

<COLUMN name="FormOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData/@FormOID</XPATH>
</COLUMN>

<COLUMN name="UserOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>
   /ODM/ClinicalData/SubjectData/StudyEventData/FormData/AuditRecord/UserRef/@UserOID
   </XPATH>
</COLUMN>

<COLUMN name="LocationOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
      <XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/AuditRecord/LocationRef/@LocationOID
      </XPATH>
```

```
   </COLUMN>

   <COLUMN name="DateTime">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData/AuditRecord/DateTimeStamp
      </XPATH>
   </COLUMN>

</TABLE>

<!--  +=========================================================+ -->
<!--  + Signature                                               + -->
<!--  +=========================================================+ -->
<TABLE name="Signature">
<TABLE_DESCRIPTION>Signature</TABLE_DESCRIPTION>
   <TABLE_XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData</TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/@StudyOID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/@MetaDataVersionOID</XPATH>
   </COLUMN>

   <COLUMN name="SubjectKey" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/SubjectData/@SubjectKey</XPATH>
   </COLUMN>

   <COLUMN name="TransactionType" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/SubjectData/@TransactionType</XPATH>
   </COLUMN>

   <COLUMN name="StudyEventOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/@StudyEventOID</XPATH>
   </COLUMN>

   <COLUMN name="FormOID" retain="yes">
```

```
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData/@FormOID</XPATH>
   </COLUMN>

   <COLUMN name="UserOID">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/Signature/UserRef/@UserOID
      </XPATH>
   </COLUMN>

   <COLUMN name="LocationOID">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/Signature/LocationRef/@LocationOID
      </XPATH>
   </COLUMN>

   <COLUMN name="DateTime">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>40</LENGTH>
      <XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/Signature/DateTimeStamp
      </XPATH>
   </COLUMN>

</TABLE>

<!--  +=============================================================+ -->
<!--  + ItemData                                                    + -->
<!--  +=============================================================+ -->
<TABLE name="ItemData">
<TABLE_DESCRIPTION>ItemData</TABLE_DESCRIPTION>
   <TABLE_XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/ItemGroupData/ItemData
   </TABLE_XPATH>

   <COLUMN name="StudyOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
      <XPATH>/ODM/ClinicalData/@StudyOID</XPATH>
   </COLUMN>

   <COLUMN name="MetaDataVersionOID" retain="yes">
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>20</LENGTH>
```

```
   <XPATH>/ODM/ClinicalData/@MetaDataVersionOID</XPATH>
</COLUMN>

<COLUMN name="SubjectKey" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/@SubjectKey</XPATH>
</COLUMN>

<COLUMN name="TransactionType" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/@TransactionType</XPATH>
</COLUMN>

<COLUMN name="StudyEventOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/@StudyEventOID</XPATH>
</COLUMN>

<COLUMN name="FormOID" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>/ODM/ClinicalData/SubjectData/StudyEventData/FormData/@FormOID</XPATH>
</COLUMN>

<COLUMN name="ItemGroupRepeatKey" retain="yes">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>8</LENGTH>
   <XPATH>
      /ODM/ClinicalData/SubjectData/StudyEventData/FormData/ItemGroupData/@ItemGroupRepeatKey
   </XPATH>
</COLUMN>

<COLUMN name="ItemOID">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>20</LENGTH>
   <XPATH>
   /ODM/ClinicalData/SubjectData/StudyEventData/FormData/ItemGroupData/ItemData/@ItemOID
   </XPATH>
</COLUMN>

<COLUMN name="Value">
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>200</LENGTH>
   <XPATH>
   /ODM/ClinicalData/SubjectData/StudyEventData/FormData/ItemGroupData/ItemData/@Value
```

```
         </XPATH>
      </COLUMN>

   </TABLE>

   <!--  +=========================================================+ -->
   <!--  + CodeListItem                                            + -->
   <!--  +=========================================================+ -->
   <TABLE name="CodeListItem">
   <TABLE_DESCRIPTION>CodeListItem</TABLE_DESCRIPTION>
      <TABLE_XPATH>/ODM/Study/MetaDataVersion/CodeList/CodeListItem</TABLE_XPATH>

      <COLUMN name="StudyOID" retain="yes">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>20</LENGTH>
         <XPATH>/ODM/Study/@OID</XPATH>
      </COLUMN>

      <COLUMN name="MetaDataVersionOID" retain="yes">
         <TYPE>character</TYPE>
         <DATATYPE>STRING</DATATYPE>
         <LENGTH>20</LENGTH>
         <XPATH>/ODM/Study/MetaDataVersion/@OID</XPATH>
      </COLUMN>

      <COLUMN name="CodeListOID" retain="yes">
         <type> character </type>
         <DATATYPE>string</DATATYPE>
         <LENGTH>20</LENGTH>
         <XPATH>/ODM/Study/MetaDataVersion/CodeList/@OID</XPATH>
      </COLUMN>

      <COLUMN name="CodedValue">
         <type> character </type>
         <DATATYPE>string</DATATYPE>
         <LENGTH>80</LENGTH>
         <XPATH>/ODM/Study/MetaDataVersion/CodeList/CodeListItem/@CodedValue</XPATH>
      </COLUMN>

      <COLUMN name="Decode">
         <type> character </type>
         <DATATYPE>string</DATATYPE>
         <LENGTH>80</LENGTH>
         <XPATH>/ODM/Study/MetaDataVersion/CodeList/CodeListItem/Decode/TranslatedText</XPATH>
      </COLUMN>
   </TABLE>
</SXLEMap>
```

EXAMPLE 3: SAS CODE TO TRANSFORM THE DATA

```
%let BYVARS = StudyEventOID SubjectKey ItemGroupRepeatKey;
%let INNER = ItemGroupRepeatKey;
%let ITEMDATA = o.ItemData;
```

```
proc sql;
  reset noprint;
  select distinct FormOID into: FORMS separated by ' '
  from o.ItemGroupRef;

  select distinct ItemGroupOID into: TABLES separated by ' '
  from o.ItemRef;

  create table ItemGroupRef as
  select
    D.ItemGroupOID,
    D.Name,
    D.SASDataSetName,
    R.FormOID
  from o.ItemGroupRef R, o.ItemGroupDef D
  where R.ItemGroupOID = D.ItemGroupOID;

create table ItemRef as
  select
    D.ItemOID,
    D.Name,
    D.DataType,
    D.SASFieldName,
    R.ItemGroupOID
  from o.ItemRef R, o.ItemDef D
  where R.ItemOID = D.ItemOID;

  quit;
  run;

%macro MakeTOC(TABLES=);
%local C T;
%let C = 1;
%let T = %scan( &TABLES, &C, ' ' );
%do %while (&T ^=);
  %global I_&T N_&T F_&T L_&T S_&T B_&T;
  proc sql;
  reset noprint;
    /* ItemOID as the I list */
    select ItemOID into: I_&T separated by ' '
    from ItemRef
    where ItemGroupOID = "&T."
    ;
     /* Field names as the N list */
    select SASFieldName into: N_&T separated by ' '
    from ItemRef
    where ItemGroupOID = "&T."
    ;
     /* Field types as the F list */
    select DataType into: F_&T separated by ' '
    from ItemRef
    where ItemGroupOID = "&T."
    ;
     /* Field labels as the L list */
```

```
    select Name into: L_&T separated by '#'
    from ItemRef
    where ItemGroupOID = "&T."
    ;
     /* Table labels as the B list */
    select Name into: B_&T
    from o.ItemGroupDef
    where ItemGroupOID = "&T."
    ;
     /* Table name as the S list */
    select SASDataSetName into: S_&T
    from o.ItemGroupDef
    where ItemGroupOID = "&T."
    ;
  quit;
  %let C = %eval( &C + 1 );
  %let T = %scan( &TABLES, &C, ' ' );
%end;
%mend MakeTOC;
%macro MakePages(FORMS=);
%local C P;
%let C = 1;
%let P = %scan( &FORMS, &C, ' ' );
%do %while (&P ^=);
  %global IG_&P DSN_&P;
  proc sql;
  reset noprint;
    /* ItemGroupOID as the IG list */
    select ItemGroupOID into: IG_&P separated by ' '
    from ItemGroupRef
    where FormOID = "&P."
    ;
    /* Data Set Name as the DSN list */
    select SASDataSetName into: DSN_&P separated by ' '
    from ItemGroupRef
    where FormOID = "&P."
    ;
  quit;
  %let C = %eval( &C + 1 );
  %let P = %scan( &FORMS, &C, ' ' );
%end;
%mend MakePages;
%macro xAssign(OID=,NAM=,FMT=);
%local C I F N;
%let C = 1;
%let I = %scan( &OID, &C, ' ' );
%do %while ( &I ^= );
  %let N = %scan( &NAM, &C, ' ' );
  %let F = %scan( &FMT, &C, ' ' );
  if ItemOID = "&I" then do;
    %if &F = text %then %do;
      &N = trim( Value );
    %end;
    %else %if &F = date %then %do;
      &N = input( Value, yymmdd10. );
```

```
    %end;
    %else %do;
      &N = Value;
    %end;
  end;
  %let C = %eval( &C + 1 );
  %let I = %scan( &OID, &C, ' ' );
%end;
%mend xAssign;

%macro xLabel(OID=,NAM=,LAB=);
%local C I L N;
%let C = 1;
%let I = %scan( &OID, &C, ' ' );
%do %while ( &I ^= );
  %let N = %scan( &NAM, &C, ' ' );
  %let L = %scan( &LAB, &C, '#' );
  &N = "&L"
  %let C = %eval( &C + 1 );
  %let I = %scan( &OID, &C, ' ' );
%end;

%mend xLabel;

%macro xRename(OID=,NAM=,FMT=);
%local C I F N;
%let C = 1;
%let I = %scan( &OID, &C, ' ' );
%do %while ( &I ^= );
  %let N = %scan( &NAM, &C, ' ' );
  %let F = %scan( &FMT, &C, ' ' );
  %if &F ^= text %then %do;
     &N = _&N
  %end;
  %let C = %eval( &C + 1 );
  %let I = %scan( &OID, &C, ' ' );
%end;
%mend xRename;

%macro xConvert(OID=,NAM=,FMT=);
%local C I F N;
%let C = 1;
%let I = %scan( &OID, &C, ' ' );
%do %while ( &I ^= );
  %let N = %scan( &NAM, &C, ' ' );
  %let F = %scan( &FMT, &C, ' ' );
  %if &F = date %then %do;
      format &N date9.;
      length &N 8;
     &N = _&N;
  %end;
  %else %if &F = integer %then %do;
      length &N 8;
     &N = _&N;
  %end;
```

```
  %else %if &F = float %then %do;
      length &N 8;
     &N = _&N;
  %end;
  %let C = %eval( &C + 1 );
  %let I = %scan( &OID, &C, ' ' );
%end;
%mend xConvert;

%macro ReadAPage(PAGE=);
%local C DSN;
%let C = 1;
%let DSN = %scan( &&DSN_&PAGE, &C, ' ' );
%do %while( &DSN ^= );
data &DSN._&PAGE( keep = &BYVARS &&N_&DSN );
  set &ITEMDATA.( where = (
    FormOID = "&PAGE." & TransactionType = 'Insert' ) );
  by &BYVARS;
  retain &&N_&DSN;
  array cvars{*} $ 16 &&N_&DSN;
  if first.&INNER then do;
    do i = 1 to dim( cvars );
      cvars{i} = "";
    end;
  end;
  %xAssign(
    OID = &&I_&DSN,
     NAM = &&N_&DSN,
     FMT = &&F_&DSN
     );
  if last.&INNER then output;
  run;
  %let C = %eval( &C + 1 );
  %let DSN = %scan( &&DSN_&PAGE, &C, ' ' );
%end;
%mend ReadAPage;

%macro ReadPages;
%local CC P;
proc sql;
  reset noprint;
  select distinct formoid into :FORMS separated by ' '
  from o.formdef;
  quit;
%let CC = 1;
%let P = %scan( &FORMS, &CC, ' ' );
%do %while( &P ^= );
  %ReadAPage(PAGE=&P)
  %let CC = %eval( &CC + 1 );
  %let P = %scan( &FORMS, &CC, ' ' );
%end;
%mend ReadPages;
%macro MakeATable(DSN=,FORMS=);
%local C P;
data &DSN;
```

```
  set
  %let C = 1;
  %let P = %scan( &FORMS, &C, ' ' );
  %do %while( &P ^= );
    %if %sysfunc(exist(&DSN._&P)) %then &DSN._&P;
    %let C = %eval( &C + 1 );
    %let P = %scan( &FORMS, &C, ' ' );
  %end;
  ;
  run;
data &DSN.( drop = _: label = "&&B_&DSN" );
    set &DSN.( rename = (
    %xRename(
       OID = &&I_&DSN,
       NAM = &&N_&DSN,
       FMT = &&F_&DSN
    ) ) );
    label %xLabel(
      OID = &&I_&DSN,
       NAM = &&N_&DSN,
       LAB = &&L_&DSN
    )
    ;
    %xConvert(
       OID = &&I_&DSN,
       NAM = &&N_&DSN,
       FMT = &&F_&DSN
    )
    run;

%mend MakeATable;
%macro MakeTables;
%local CC D;
proc sql;
  reset noprint;
  select distinct SASDatasetName into :DATASETS separated by ' '
  from o.ItemGroupDef;
  select distinct formoid into :FORMS separated by ' '
  from o.formdef;
  quit;
%let CC = 1;
%let D = %scan( &DATASETS, &CC, ' ' );
%do %while( &D ^= );
  %MakeATable(DSN=&D,FORMS=&FORMS)
  %let CC = %eval( &CC + 1 );
  %let D = %scan( &DATASETS, &CC, ' ' );
%end;
%mend MakeTables;

%MakeTOC(TABLES=&TABLES)
%MakePages(FORMS=&FORMS)

%ReadPages
%MakeTables
```