Paper 119-29

Reading and Writing XML files from SAS®

Miriam Cisternas, Ovation Research Group, Carlsbad, CA
Ricardo Cisternas, MGC Data Services, Carlsbad, CA

ABSTRACT

XML (eXtensible Markup Language) is gaining popularity as a medium for data exchange. SAS Institute has added additional support recently for reading and writing XML files under both 8.2 and 9.1. The purpose of this workshop is to provide a quick introduction to XML, show the ways in which XML is read and written from SAS, and then present hands-on examples for reading in and mapping XML files to SAS datasets and writing out SAS datasets to XML files.

INTRODUCTION

This workshop is targeted toward SAS programmers with a working knowledge of BASE SAS and familiarity with HTML or another markup language. This workshop will guide you through a series of examples that illustrate how to read and write XML files from SAS on computers running SAS 9.1 with the stand-alone SAS XML Mapper tool version 9.1.10 installed.

THE CHALLENGE

XML is a language that is hierarchical and describes data in terms of markup tags. But SAS is usually implemented using a relational model, even though data sets are often not fully normalized. Therefore, translation between these two formats requires at least a rudimentary understanding of the structure behind the data.

Figure 1 illustrates a possible XML representation of some clinical trial specimen data. In this case, all clinical trial data are grouped under a single study which is identified by a StudyID. A study may have multiple sites from which data are collected and each site is identified by a unique SiteID. A site has multiple participants who may visit the site on multiple visits. Each visit has a date and VisitID and may include the collection of specimens. In turn, each specimen has a type and it is collected in one or more tubes, all of which share the same tube type.

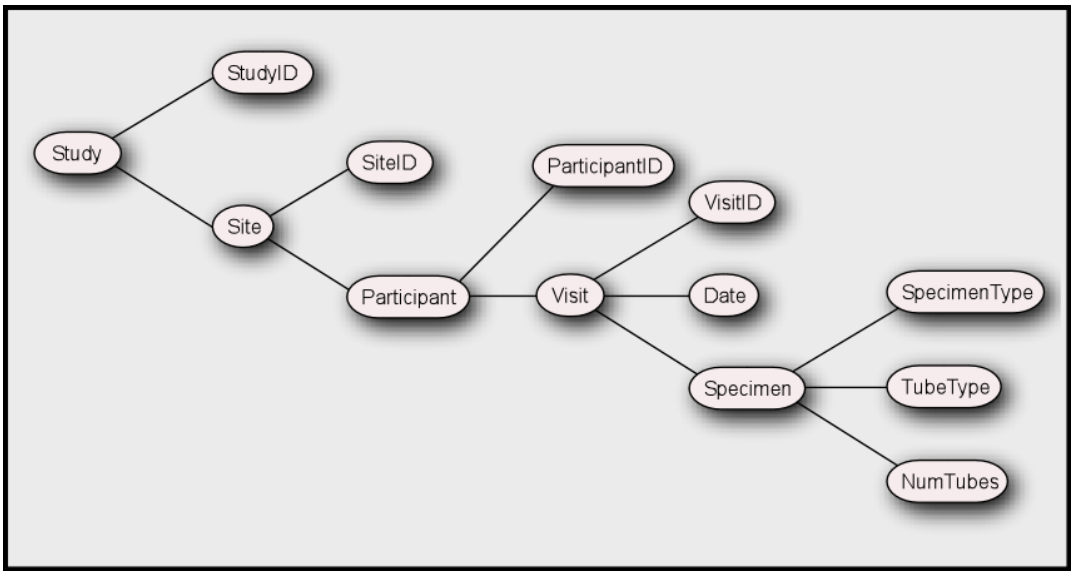

Figure 1. Possible XML (tree) representation of clinical trial specimen data

Figure 2 diagrams a possible relational view of the same data displayed in Figure 1. The relational diagram stores the same information as the hierarchical diagram in Figure 1 preserving both the data and the relationships between the data entities. In this case, the names of the tables can be extracted by determining the critical path among the nodes of the tree in Figure 1. At each level of the tree, there is a single node that acts as a root for a sub-tree (i.e. Study, Site).

Each one of these critical path nodes becomes a table in the relational diagram while the rest of the nodes become column names within the table named after its parent node. Some of these subordinate nodes become primary keys within their tables if they can uniquely identify a table entry. For instance, the table Site uses the column SiteID as a

primary key since it is a unique identifier for all site records.

The primary keys are also reproduced in related tables as foreign keys to establish relationships among tables. For example, SiteID is the primary key to the Site table and it is also a foreign key within the Participant table reflecting the relationship between Site and Participant. Additionally, the links among the tables preserve the cardinality of the relationships.

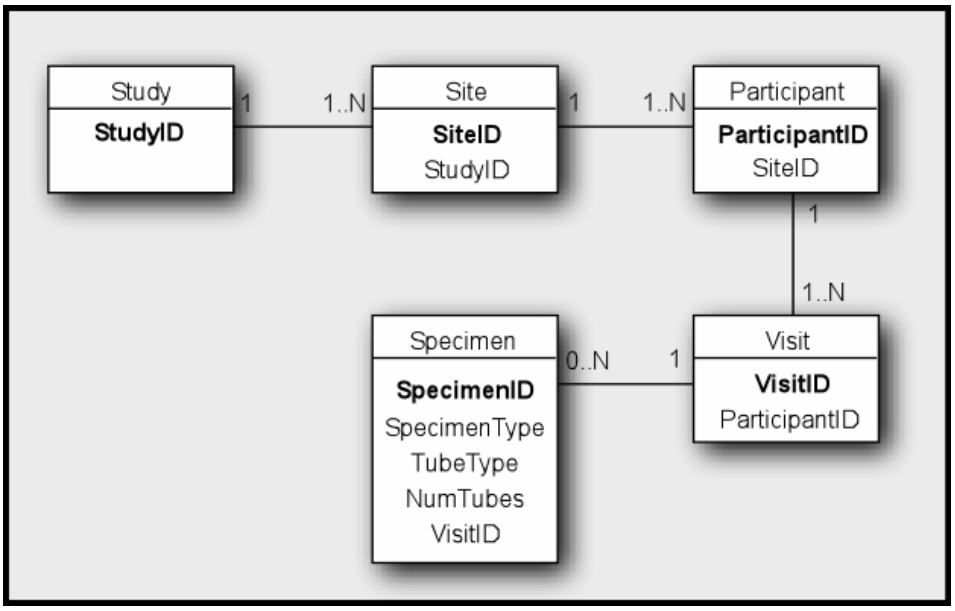

Figure 2. Possible relational representation of clinical trial specimen data from Figure 1

Transferring data from a hierarchical structure (XML) to a relational structure (SAS) is a non-trivial problem. It requires good understanding of the data to be able to identify primary keys, foreign keys, and dataset variables. Data transfer from a relational structure (SAS) to a hierarchical structure (XML) is also difficult. You need to identify which data elements take precedence in the hierarchy and establish an order relationship. In either case, the programmer must have a solid understanding of the source data to be transferred.

BASIC RULES OF XML

While SAS now has tools for you to read in XML documents without your needing to become an XML guru, it is important to familiarize oneself with the following basic rules of XML before attempting to convert XML files to SAS datasets and vice versa:

- The basic building block of XML is an element. Elements begin with an open tag <element_name>
- Elements can either end with an end tag </element_name> or terminate with '/>' in their open tag <element_name/>
- All tags must be fully nested – no overlapping tag boundaries are allowed.
- A tag may optionally include a value.
- A tag may optionally include attributes, which are values enclosed in double quotes.
- A comment is text delimited by '<!--' and '-->'.

ACTIVITY 1: VIEWING AN XML FILE FROM INTERNET EXPLORER

Since XML documents are text files, they can be viewed in virtually any text editor. In addition, there are XML editors available that simplify the process of writing and editing XML documents, including the SAS XML Mapper tool provided with SAS 9.1.

But even using Microsoft's Internet Explorer to view an XML file can provide us valuable information about its structure that we cannot see in a simple text editor such as NotePad. Consider the XML document displayed in the box on the following page:

```
<!DOCTYPE client_list SYSTEM "..\XML\client_list.dtd">
<client_list>
    <client status="active">
           <name>John Doe</name>
           <address>1212 Maple Road</address>
           <city>Springfield</city>
           <state>CA</state>
           <zip>91234</zip>
    </client>
    <client status="inactive">
           <name>Mary Doe</name>
           <address>1212 Maple Road</address>
           <city>Springfield</city>
           <state>CA</state>
           <zip>91234</zip>
    </client>
    <client status="active">
           <name>John Public</name>
           <address>100 Byron Road</address>
           <city>Carlsbad</city>
           <state>CA</state>
           <zip>99999</zip>
    </client>
    <client status="active">
           <name>Fionnula Jackson</name>
           <address>444 First Street</address>
           <city>San Mateo</city>
           <state>CA</state>
           <zip>94402</zip>
    </client>
</client_list>
```

If the file containing the statements in the box above is named C:\workshop\ws119\XML\client_list.xml, you can bring the file into Internet Explorer as follows:

- From **Windows Explorer** or **My Computer**, navigate to the C:\workshop\ws119\XML subdirectory.
- Right-click on client_list.xml.
- Select **Open With** and **then Internet Explorer**.

The file will be displayed as in Figure 3, right.

Internet Explorer displays XML in a tree view where each parent node is preceded by a control (a plus or minus sign) that allows the user to collapse or expand a sub-tree of nodes.

IE also color codes the display of elements and attributes (maroon), data (black), triangular brackets (blue), directives (blue), and comment text (grey).

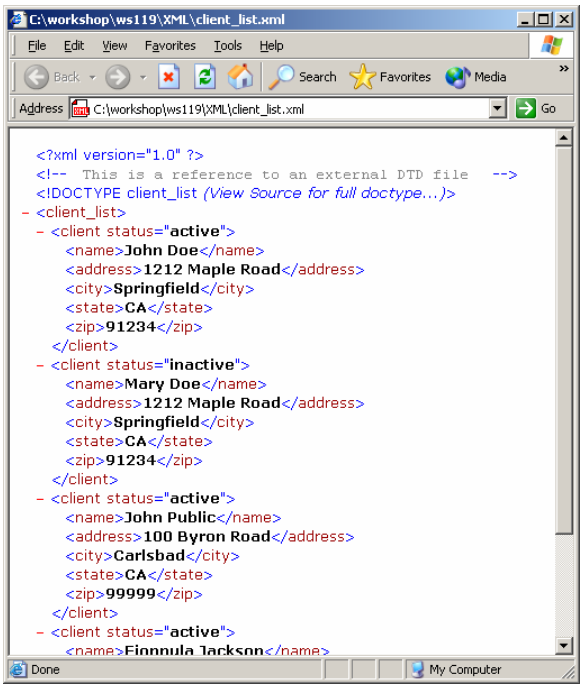

Figure 3. Internet Explorer rendering of an XML file.

ACTIVITY 2: READING AN XML DOCUMENT INTO A SAS DATASET

The easy way to access and write XML from SAS is to use the XML engine on a libname statement (versions 8.1 and above). However, using this statement without any other options requires that the XML file structure contain only rectangular datasets. In other words, the hierarchy for the source XML Document can only be three levels deep, containing a root element, second-level elements corresponding to the table to be read in, and third-level elements corresponding to the variables. For this activity, we use the XML engine to parse the incoming file called hosp_discharge.xml, which contains an excerpt of a hospital discharge data file.

The syntax of the XML libname is similar to that of a standard SAS libname, but (1) xml is placed after the libname keyword and before the location to indicate that the XML engine is to be used, and (2) the location in quotes refers to the specific XML file, not just a directory path.

Before reading in an XML file into SAS, it is a good idea to examine it. Figure 4 shows a screenshot of the hosp_discharge.xml file in Internet Explorer. The file contains a root-level node called <hosp_discharge> which in turn contains a number of <discharge> elements. Each <discharge> element has 5 children elements <patient_ID>, <SEX>, <admit_date>, <DOB>, and <discharge_date>.

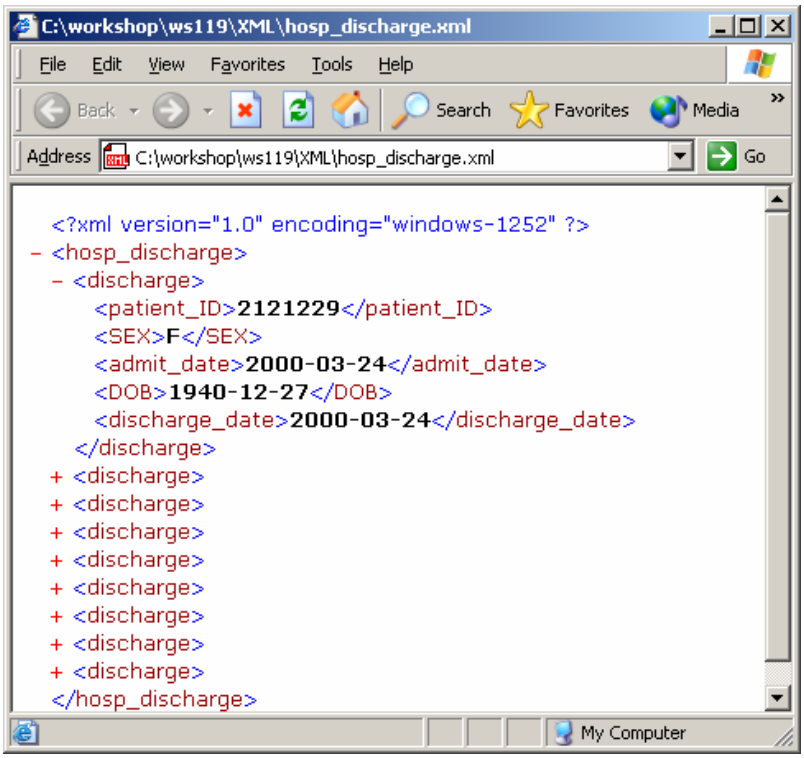

Figure 4. Display of rectangular XML file to be read into SAS

The SAS code to read the XML file is as follows:

```
title1 'XML WORKSHOP EXERCISE #2';
libname in xml 'C:\workshop\ws119\XML\hosp_discharge.xml';
data discharge;
  set in.discharge;
run;

title2 'check parsing of file';
proc print data=discharge;
run;

proc contents data=discharge;
run;
```

The PROC PRINT of the resulting SAS dataset is reproduced below:

Obs	DISCHARGE_ DATE	DOB	ADMIT_DATE	SEX	PATIENT_ ID
1	2000-03-24	1940-12-27	2000-03-24	F	2121229
2	2000-03-30	1946-08-25	2000-03-29	F	3359900
3	2000-03-05	1910-06-30	2000-03-03	M	4245123
4	2000-03-28	1946-06-13	2000-03-25	M	6919113
5	2000-03-01	1941-02-01	2000-02-25	F	2836181
6	2000-12-22	1928-11-25	2000-12-20	M	7098312
7	2000-03-27	1922-09-10	2000-03-26	M	9983225
8	2000-03-09	1916-08-31	2000-03-09	F	1299349
9	2000-03-27	1946-06-05	2000-03-26	M	555856

An excerpt from the PROC CONTENTS is below:

#	Variable	Type	Len	Format	Informat	Label
3	ADMIT_DATE	Num	8	IS8601DA.	YYMMDD.	ADMIT_DATE
1	DISCHARGE_DATE	Num	8	IS8601DA.	YYMMDD.	DISCHARGE_DATE
2	DOB	Num	8	IS8601DA.	YYMMDD.	DOB
5	PATIENT_ID	Num	8	F8.	F8.	PATIENT_ID
4	SEX	Char	1	$1.	$1.	SEX

If we examine the SAS code and the XML file on the previous page we notice that the name of the SAS dataset is the same as the name of the second-level element in the XML file (<discharge>). Additionally, the variable names in the PROC PRINT output match the name of the third-level elements in the XML file. However, the case of the element names is not preserved, as SAS converts all characters in a name to upper case. Finally, the XML engine formats dates using the IS8601DA. (yyyy-mm-dd) format. (Former releases transformed such dates to character fields.)

ACTIVITY 3: CONVERTING A SAS DATASET TO XML

To convert a SAS dataset to XML, create an XML libname and then write to it. The following lines of code illustrate how to write out the dataset created in activity #2 to an XML file. The program adds in a SAS date variable, xfer_date, to record the date we transferred the file back to XML format.

```
title1 'XML WORKSHOP EXERCISE #3';
libname out xml 'C:\workshop\ws119\XML\discharge_from_SAS.xml';

data out.discharge;
  set discharge;
  ***let's add in the date converted as today's date and format it in ISO8601
  ***(yyyy-mm-dd) format;
  xfer_date=today();
  format xfer_date is8601da10.;
run;
```

The resulting XML file as seen through Internet Explorer (partially collapsed) is displayed in Figure 5, below. Notice the following about the resulting XML file in Figure 5:

- The root element is <TABLE>. This is a convention SAS uses when writing out XML files using the XML libname engine.
- Tags for all the original fields are all caps (this conversion was done during the original parsing in Activity #2).
- The tags for the xfer_date field are in lower case, since we created that variable using lower case in SAS.
- All the SAS date fields are displayed in SAS internal numeric data storage (# of days since January 1, 1960). This is because formats are not preserved in general when copying to non-native engines such as XML. In fact, your SAS log should show the following note:

```
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
```

However, in 9.1, SAS dates formatted with the DATEw. format are mapped to ISO 8601 representation. Thus one solution to this problem is to format all the date variables with the DATEw. format. Please note that the default behaviors of the XML engine have been changing with successive re-releases, so you should test your code carefully when reading/writing XML files in a production environment.

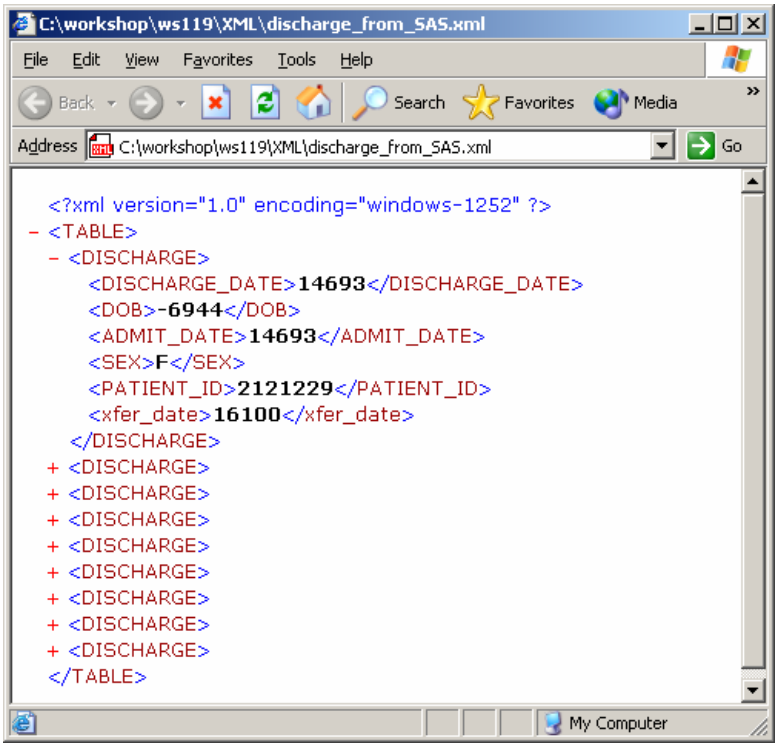

Figure 5. XML file created by writing to the XML libname engine.

ACTIVITY 4: CREATING XML FILES FROM SAS PROCS USING ODS

ODS can render output in a number of different formats such as listing, html, and rtf. One of the output destinations is XML. Following is code to render a PROC PRINT of the discharge data to an XML file.

```
ods xml    file='C:\workshop\ws119\XML\discharge_print.xml';
proc print noobs data=discharge;
run;
ods xml close;
```

If you examine the resulting file, discharge_print.xml in IE, you will notice that it follows a very rigid data structure that specifies every detail of the PROC PRINT issued including the title, the name of the data set, the descriptions of every column, and the data of the entire table on a row by row basis. This XML format carries a lot of detail which is unrelated to the data being presented. This makes it impractical as a data transport mechanism because its verbose nature makes the resulting XML files quite large and unwieldy.

ACTIVITY 5: READING IN MORE COMPLICATED XML FILES

As mentioned earlier, using the XML libname to read in XML files to SAS requires that the file be structured in a rectangular (non hierarchical) manner. Specifically, this means that each observation to be read into SAS must be appear within a second-level element, and each variable value must appear as a third-level element.

By default, XML attributes are dropped in the conversion process, and SAS determines the data type, format, and informat of all variables. SAS 9.1 includes a production version of the XMLMAP option, which can read in XML files that don't map directly to rectangular data. The SAS XML Libname engine shipped with SAS 8.2 cannot handle non-rectangular input files. Backporting of the functionality of the SAS 9.1 XML Libname engine to SAS 8.2 is expected soon.

USING THE XMLMAP OPTION REQUIRES THE CREATION OF A MAP FILE

The map file is an XML file that specifies how SAS is to map the source XML document to specific datasets, variables and observations. Documentation for this option can be found at http://support.sas.com/91doc/docMainpage.jsp. See also Anthony Friebel's paper presented at SUGI 28 which includes an extensive discussion of the XMLMAP engine (in the References section at the end of this paper).

SAS 9.1 also includes a production version of SAS XML Mapper, a stand-alone application that can be used as an XML Editor or to create XML map files with a drag and drop interface. SAS XML Mapper is located on the 9.1 client-component CDs. SAS XML Mapper should also be available soon via web download. Contact XMLEngine@sas.com for details.

For this workshop, we use XMLMAP v1.2 syntax. The structure of the basic XMLMAP schema is displayed in Figure 6 and described further below:

- The root element of the XMLMAP file is the <SXLEMAP> element. It can have one or more <TABLE> children elements. Each <TABLE> element identifies a SAS dataset by name using the name= attribute.
- A <TABLE> element has a single <TABLE-PATH> child element which specifies the point in the XML document where the table data begins. A <TABLE> element also has one or more <COLUMN> elements that identify the columns of the dataset. A <COLUMN> element will have several children:
  - ✓ a <DATATYPE> element that specifies the XML data type of the source data
  - ✓ an optional <FORMAT> element that specifies a format to be associated with the variable
  - ✓ an optional <INFORMAT> element that specifies an informat to be associated with the variable
  - ✓ a <LENGTH> element that specifies the data length
  - ✓ a <TYPE> element that indicates the SAS data type (character or numeric) used for the column
  - ✓ a <PATH> element that specifies where to look for column data within the XML input file's hierarchy

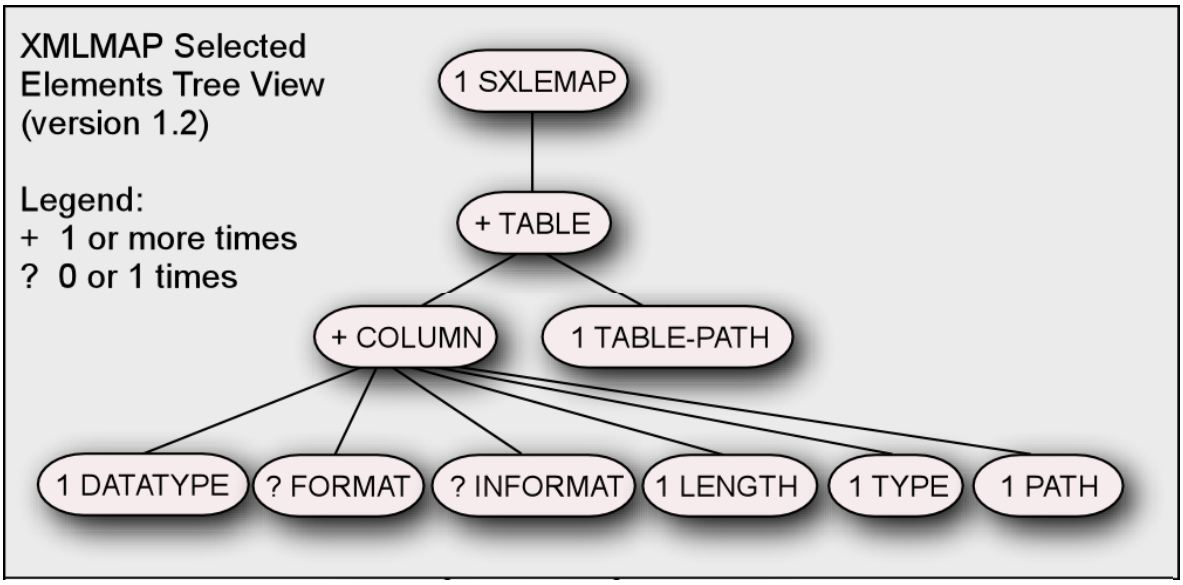

Figure 6. Selected elements of the XMLMAP Syntax v 1.2

THE EXERCISE

Use the XMLMAP option to read all client_list.xml elements (viewed in Activity 1) and map (1) the status= attribute to a character variable named status and (2) the zipcode element to a character variable.

In general, reading in more complicated XML files into SAS can be accomplished using a five-step process:

1. View the Data/Understand the structure.
2. Decide what you want to extract.
3. Use SAS XML Mapper to create the map file.
4. Use SAS XML Mapper to generate the SAS code to read in the file and reference the map file.
5. Submit your program and check that the conversion worked.

STEP 1: VIEW THE DATA AND UNDERSTAND THE STRUCTURE

First, we launch SAS XML Mapper by selecting **All Programs** from the **Start** button in Windows and picking the "SAS" Program Group. Within it, click on the **SAS XML Mapper** item to launch the application.

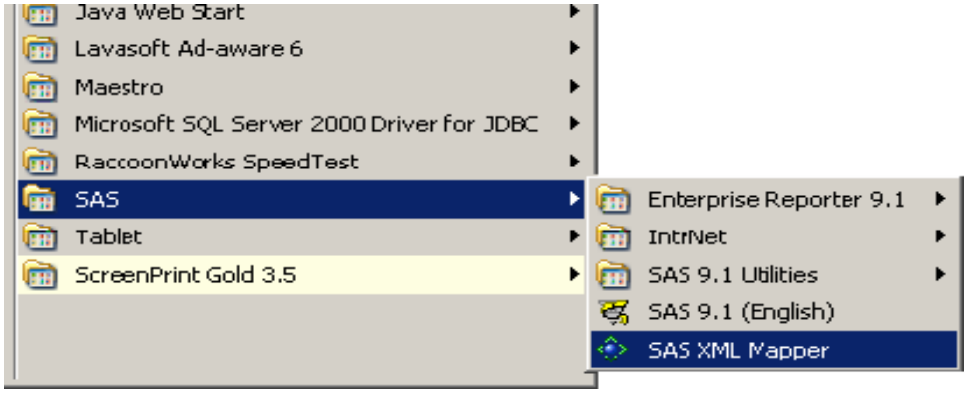

Figure 7. Launching SAS XML Mapper

The XML Mapper user interface uses a single window divided in 3 panes (see Figure 8, below). The top-left pane is named the **XML pane** and displays the tree structure of an XML file. The top-right pane is named the **XML Map pane** and displays a graphical representation of the table and columns specified for the target SAS dataset(s) and also allows the user to specify the properties and formats of each dataset column. The bottom pane is named the **Source Pane** and includes multiple tabs which in turn display the XML document source code, the source code for the automatically generated XML Map, the automatically generated SAS code used to read the XML file, a Map file validation test pane, and a Log file.

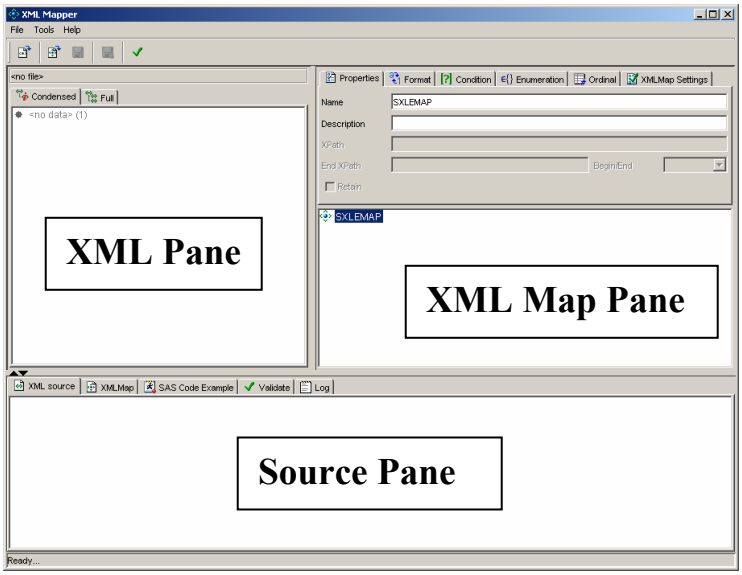

Figure 8. The SAS XML Mapper Environment

The normal sequence of actions in XML Mapper involves opening an existing XML document, creating an XML Map from the elements of the XML document, saving the resulting XML Map file, and saving the automatically generated SAS code that reads the XML document.

To open an XML document simply select the "**Open XML…**" menu item from the **File** menu, then select the name of the file to open by navigating through the directory structure. If we open the document client_list.xml, the XML Mapper interface appears as in Figure 9:

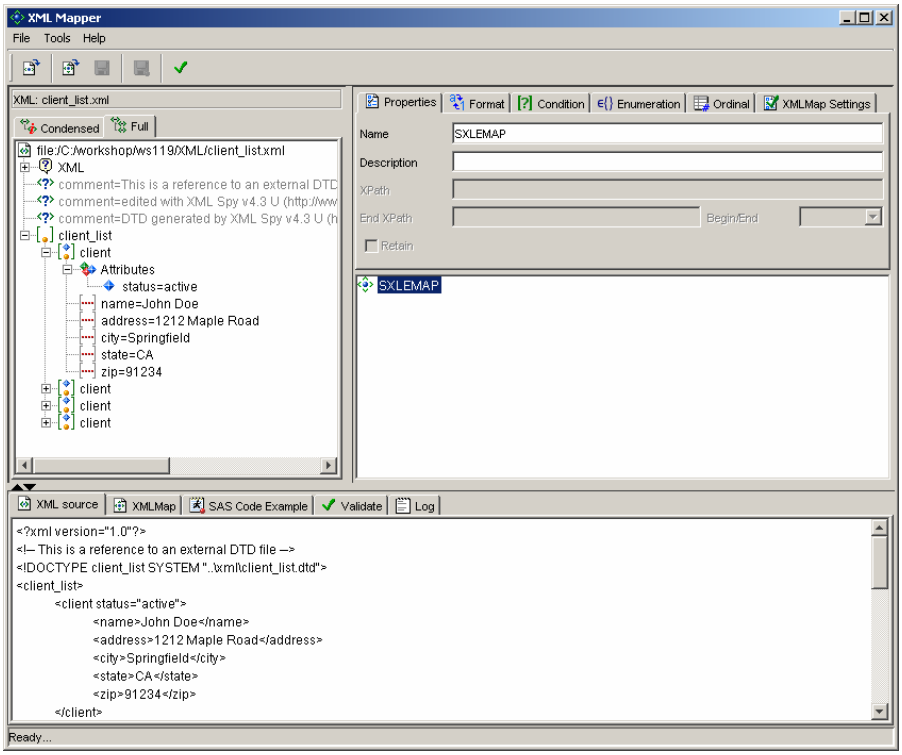

Figure 9. Displaying client_list.xml in SAS XML Mapper

STEP 2: DECIDE WHAT YOU WANT TO EXTRACT

We want to extract all the client information from the XML document. Specifically, we want the name, address, city, state, zip sub-elements of the client element as well as the active attribute from the client element. This means that we will use the client element of the XML document as the basis for our target dataset, and its sub-elements and attribute will make up the variables of the target dataset.

STEP 3: USE SAS XML MAPPER TO CREATE THE MAP FILE

It is time to create the map file. The **XML Pane** displays a tree view of the XML document where each node or attribute is a node in the tree. At the same time, the **Source Pane** displays the text of the XML document while **the XML Map Pane** is initially empty. To build an XML Map file we must drag the elements that we intend to read in from **the XML Pane** into the **XML Map Pane**. The first step is to drag the element that will constitute the basis for our entire dataset. In the case of the client_list.xml file the element that makes the logical choice for a table name is the <client> element.

To select the client element simply click and drag it from the **XML Pane** into **the XML Map Pane** dropping it over the SXLEMAP node. This action creates a table node in the XML Map pane view with the name "client". The elements and attributes selected for the dataset columns must be dragged and dropped over the "client" node in the XML Map **Pane**. The resulting tree is displayed in Figure 10. Notice how, the **Source Pane** displays the code of the automatically generated map file.

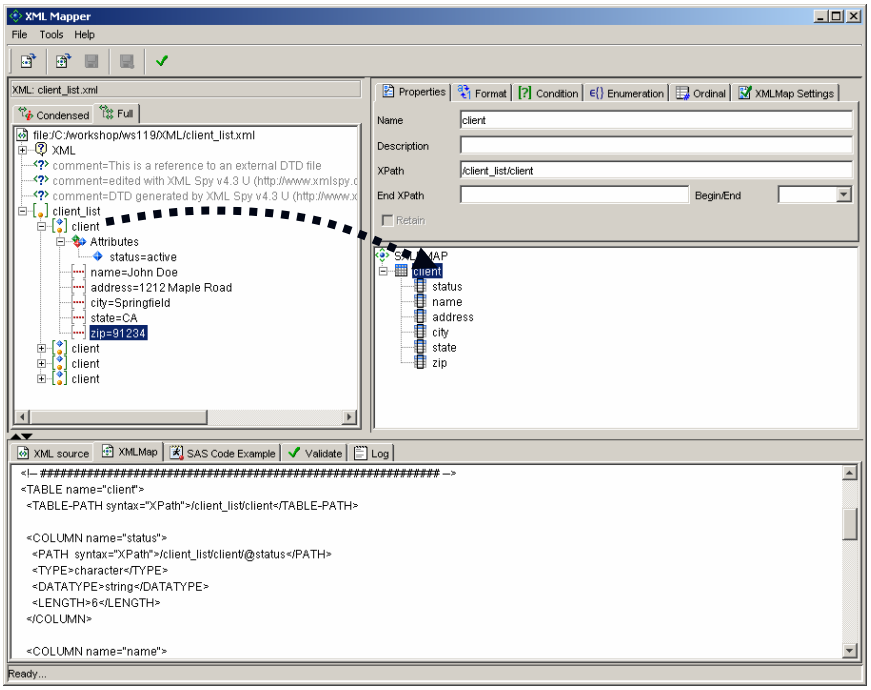

Figure 10. Creating the map file by dragging elements to the XML Map Pane

Finally, we change the default name of the XML Map (SXLEMAP) to a name meaningful to our program by clicking on the SXLEMAP node of the **XML Map Pane** and entering a new name in the **Properties** tab. A partial listing of the generated XML Map file can be seen in Figure 11.

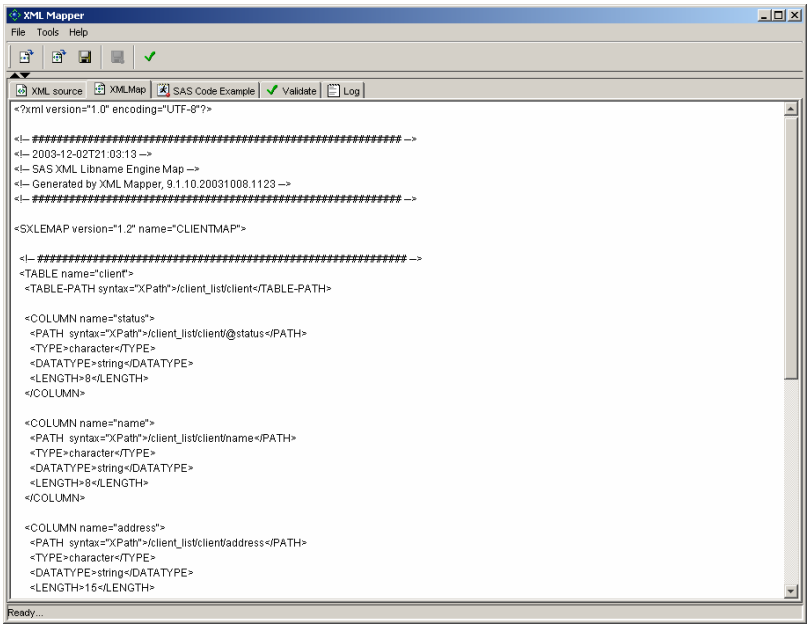

Figure 11. The map file generated by SAS XML Mapper

We then save the XML Map file by selecting the **Save XMLMap As…** item from the **File** menu.

STEP 4: USE SAS XML MAPPER TO GENERATE THE SAS CODE TO READ IN THE FILE AND REFERENCE THE MAP FILE

After we save the XML Map file, the automatically generated SAS code in the **SAS Code Example** tab of the **Source Pane** will be complete (see Figure 12). The SAS code can be saved to a .sas file by selecting the **Save SAS As…** item from the **File** menu. As you can see in Figure 12, the example code references the .map file we created in the XML libname xmlmap= option. The SAS code includes a PROC CONTENTS and PROC PRINT on a subset of observations. The PROC CONTENTS and PROC PRINT are excellent tools to check that the source data was mapped properly.

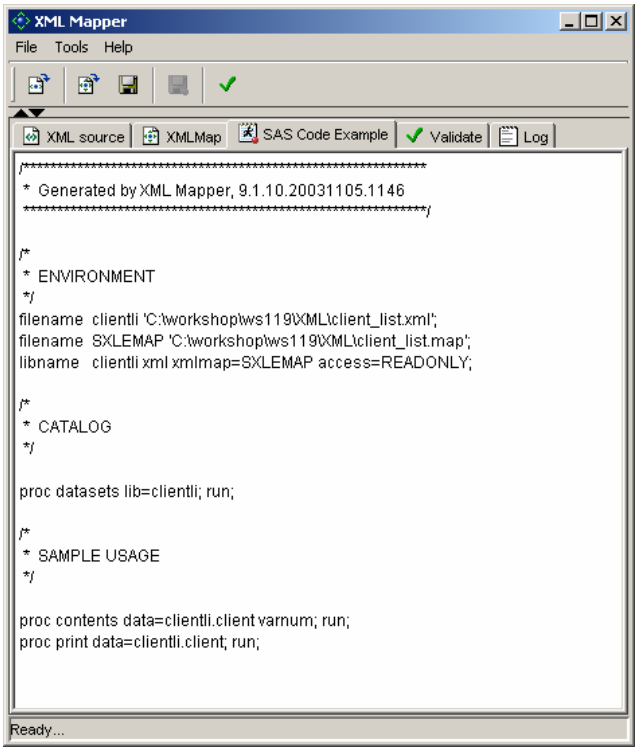

Figure 12. The SAS program generated by SAS XML Mapper

STEP 5: SUBMIT YOUR PROGRAM AND CHECK THAT YOUR CONVERSION WORKED

An excerpt from PROC CONTENTS on the converted file appears below. We successfully read in all elements as well as the status attribute, and mapped the zip element to a character variable.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
3	address	Char	16	$16.	$16.	address
4	city	Char	11	$11.	$11.	city
2	name	Char	16	$16.	$16.	name
5	state	Char	2	$2.	$2.	state
1	status	Char	8	$8.	$8.	status
6	zip	Char	5	$5.	$5.	zip

ACTIVITY 6 (EXTRA CREDIT)

In this activity, we will go through the process of reading a complex XML file (acc.xml) in more detail. We will use the same five-step process described in the previous activity.

STEP 1: VIEW THE DATA AND UNDERSTAND THE STRUCTURE

If you examine the file acc.xml in SAS XML Mapper (Figure 13) you will see that it contains lab data for a clinical trial. There is a root element called <study> that contains a variable number of children members called <participant> which in turn can have multiple children called <visit>.

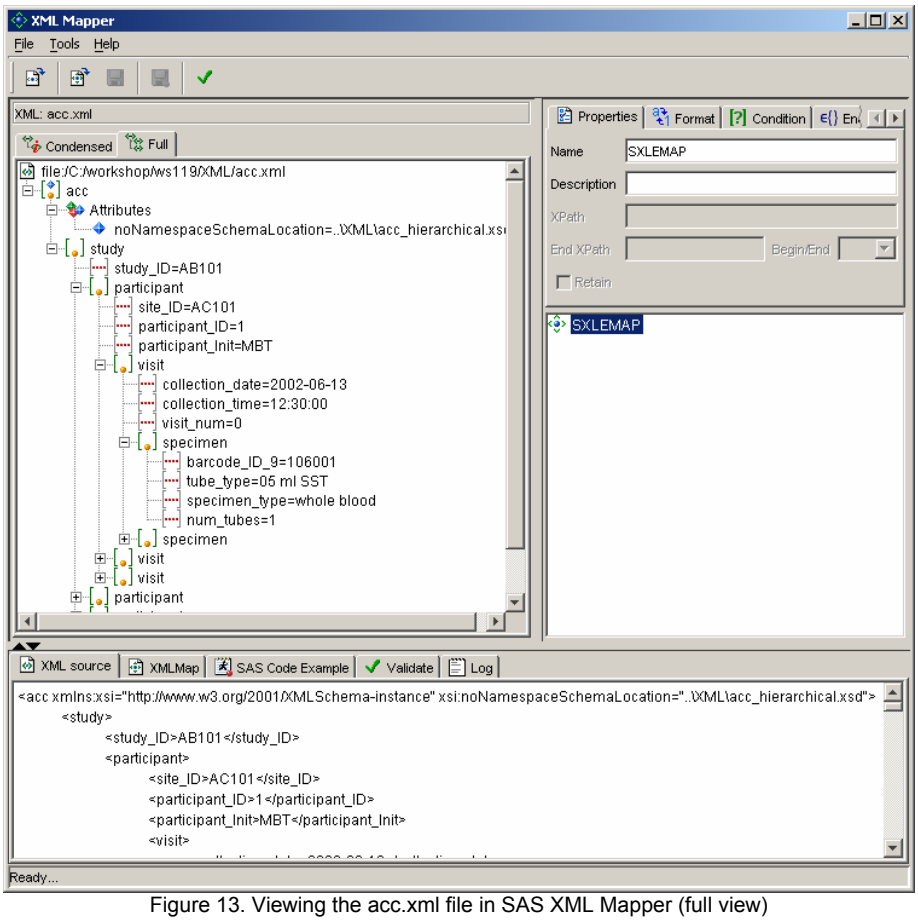

Figure 13. Viewing the acc.xml file in SAS XML Mapper (full view)

Since this hierarchy is several levels deep, it is a good idea to select the **Condensed** view tab in the **XML Pane** as shown in Figure 14. This allows us to view the document metadata and get an understanding of the hierarchy. It even shows us in the **Condensed** view tab how many times each element occurs.

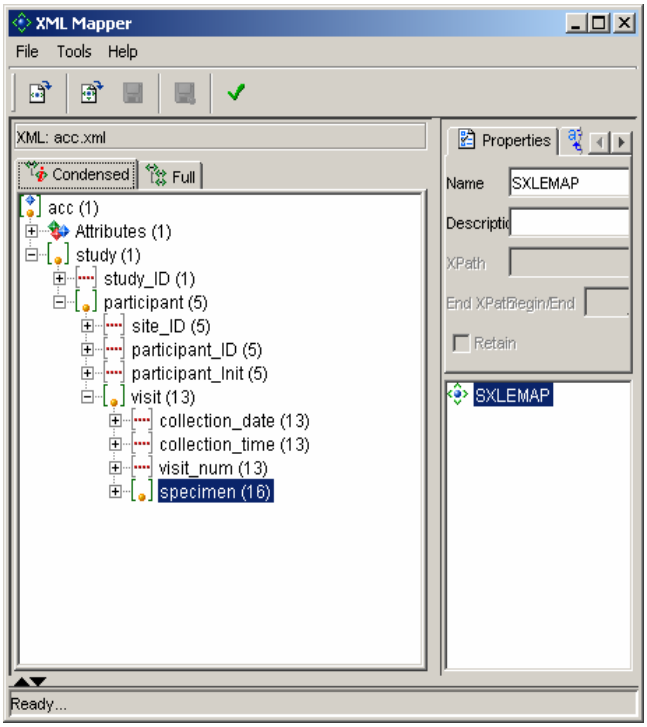

Figure 14. Viewing the acc.xml file in SAS XML Mapper (condensed view)

STEP 2: DECIDE WHAT YOU WANT TO EXTRACT

Assume that we want to extract the data from the following elements into one dataset: study_ID, site_ID, participant_ID, collection_date, collection_time, visit_num, barcode_ID_9, tube_type, and num_tubes. Further assume that the new dataset will be called “acc” and will originate in the specimen element of the XML document. We will therefore expect the dataset acc to have 16 observations.

STEP 3: USE SAS XML MAPPER TO CREATE THE MAP FILE

Since our dataset observations originate in the specimen element, we select and drag a specimen element from the **XML Pane** into the **XML Map Pane** (Figure 15). The specimen element of the XML file becomes the name of the TABLE element in the XML map (examine the **XMLMap** tab of the **Source Pane**).

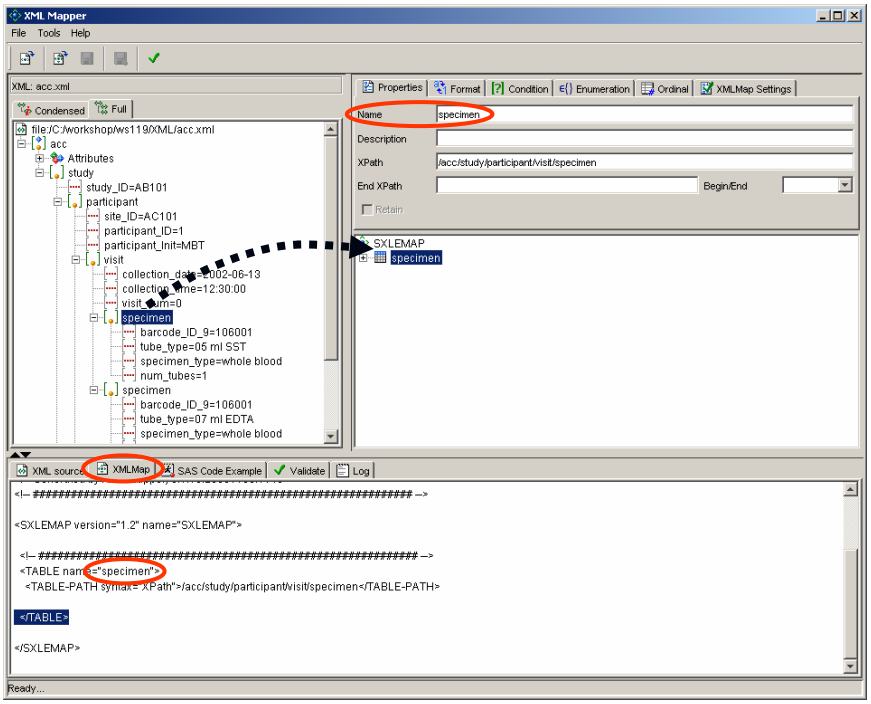

Figure 15. Creating the table via drag and drop in SAS XML Mapper

In order to have the resulting dataset member name be acc, we now change the name of the TABLE element to acc using the text box **Name** in the **Properties** tab of the **XML Map Pane** (see Figure 16).

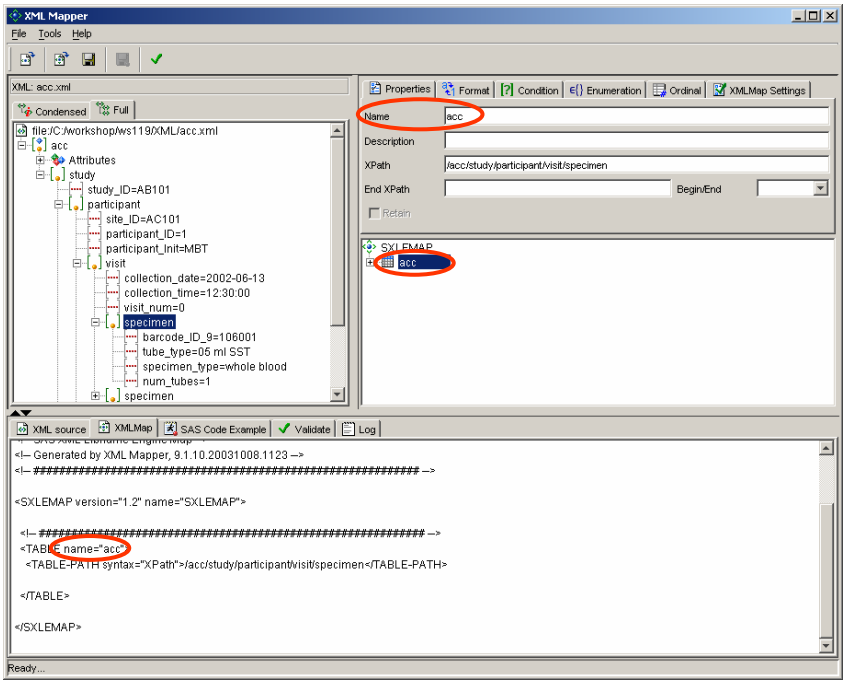

Figure 16. Renaming the table to acc using the Properties tab of the XML Map Pane

Now it's time to drag and drop the elements and attributes that will be used to extract the data for the dataset's variables maintaining their desired order. Drag each element/attribute into the acc TABLE element of the **XML Map Pane** (Figure 17). If the resulting list of columns is not in the correct order, right-click on the column name and select "Move up" or "Move Down" from the popup menu. The .map file is generated for you as you drag and drop.

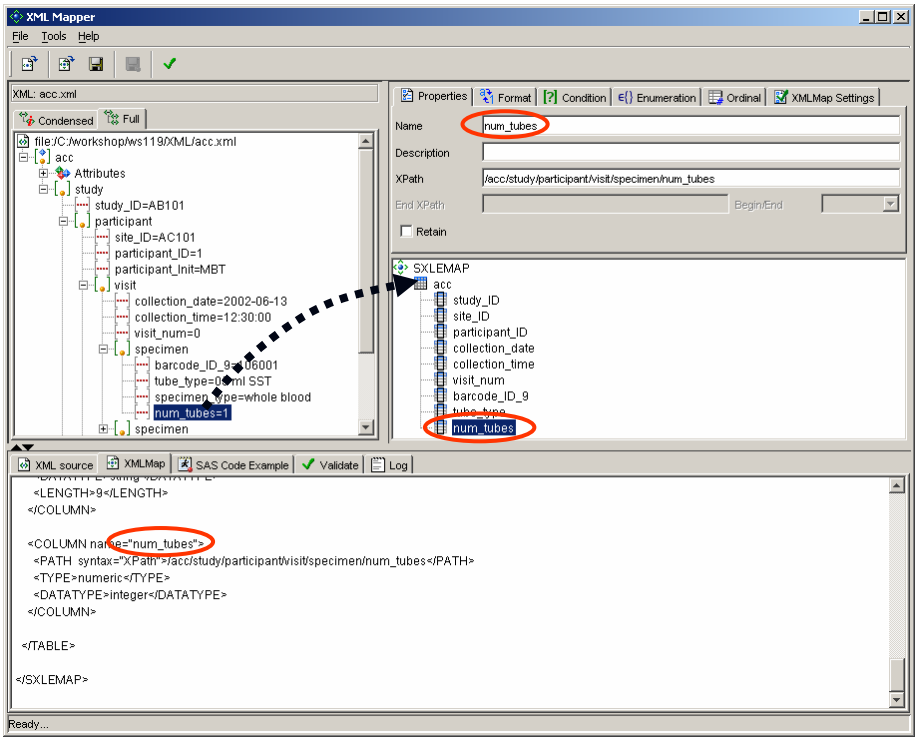

Figure 17. Dragging elements that will map to SAS variables to the XML Map Pane

If you look carefully as you drag and drop, you will observe that the XML Mapper application automatically sets the retain= attribute of a COLUMN element, if necessary. For example, the study_ID COLUMN originates in an XML element that is an ancestor of the specimen element used as the basis of the TABLE. XML Mapper automatically

checks the **Retain** box in the **Properties** tab, thereby setting the retain attribute of the study_ID COLUMN to “YES” for the generated map file, as in Figure 18, below.

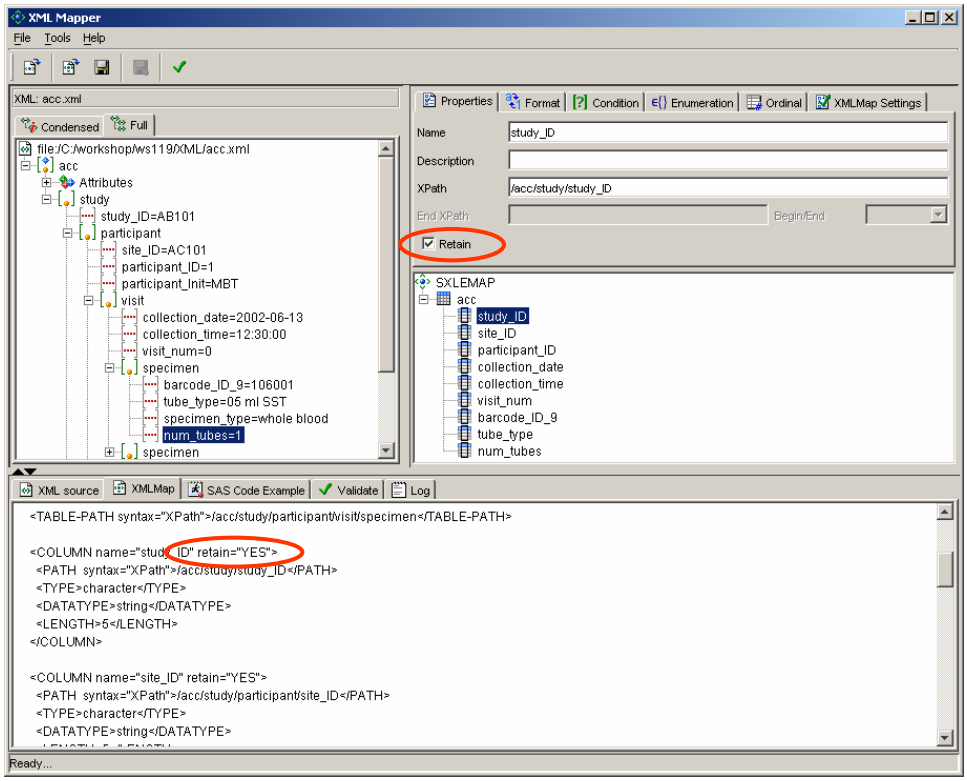

Figure 18. SAS XML Mapper anticipates when certain element values are to be retained

We are almost done with the mapping process. But we need to examine the selections for **Type**, **Datatype**, **Format** and **Informat** values for every COLUMN in the XML Map to ensure that they are what we want. We do that using the **Format** tab of the **XML Map Pane**. The XML Mapper application does a great job at guessing the desired formats for every column. For example, the collection_date column is automatically associated with a Datatype “date” and a Format “IS8601DA10.” which is equivalent to YYYY-MM-DD. See Figure 19 for an illustration.

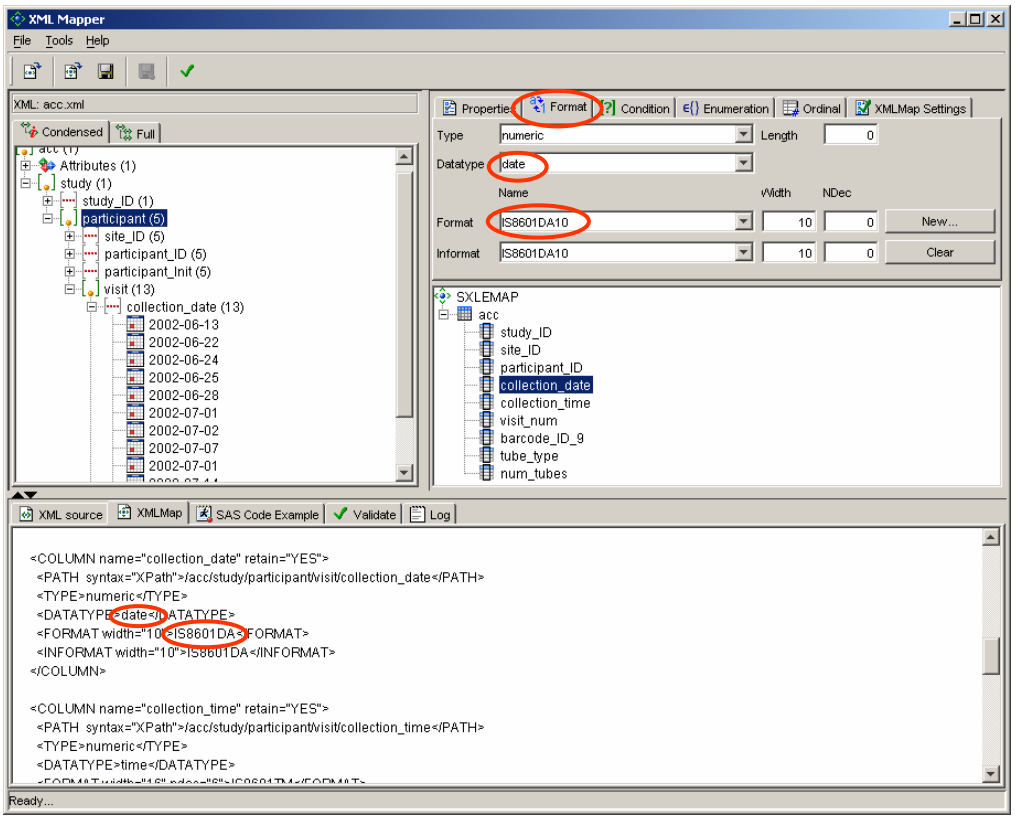

Figure 19. Checking SAS XML Mapper default format and data type selections for the collection_date variable

STEPS 4-5: CREATING AND SUBMITTING THE SAS CODE

As in Exercise #5, in order for the map validation process to work properly we must change the name of the of the SXLEMAP element. The **XML Map Pane** and **Source Pane** are updated automatically. We can now save the XML Map file using the **Save XMLMap As…** item in the File menu, naming the file accmap.map. After saving the XML Map file, the **SAS Code Example** tab in the **Source Pane** will contain the correct file names (Figure 14). We simply save that code and bring it into a SAS session to execute it. The resulting PROC CONTENTS is displayed following Figure 20.

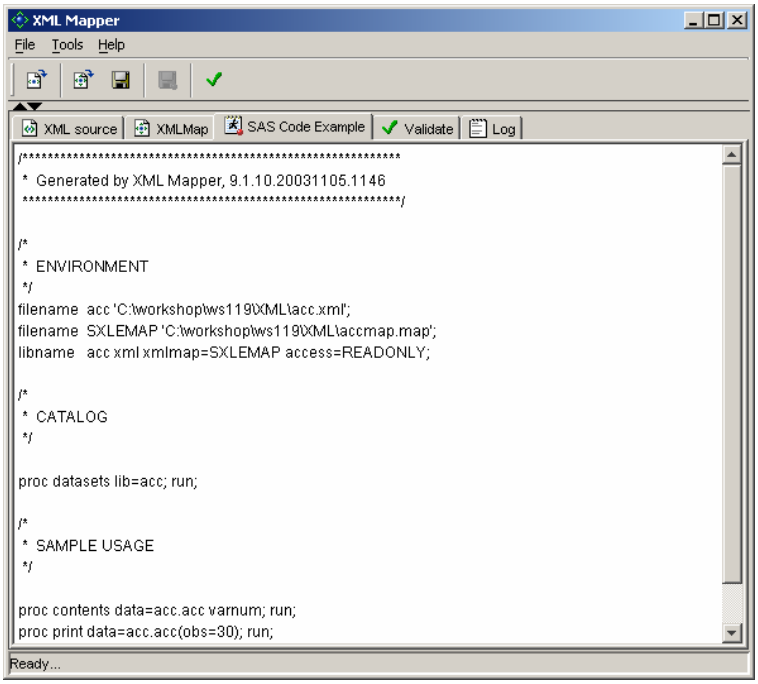

Figure 20. The SAS program generated by XML Mapper that references the map file

```
                  Data Set Name        ACC.ACC     Observations          .
                  Member Type          DATA        Variables             9
                  Engine               XML         Indexes               0
                  Created              .           Observation Length    0
                  Last Modified        .           Deleted Observations  0
                  Protection                       Compressed            NO
                  Data Set Type                    Sorted                NO
                  Label
                  Data Representation  Default
                  Encoding             Default


                              Variables in Creation Order

     #    Variable            Type  Len    Format          Informat       Label

     1    study_ID            Char    5    $5.             $5.            study_ID
     2    site_ID             Char    5    $5.             $5.            site_ID
     3    participant_ID      Num     8    F8.             F8.            participant_ID
     4    collection_date     Num     8    IS8601DA10.     IS8601DA10.    collection_date
     5    collection_time     Num     8    IS8601TM16.6    IS8601TM16.6   collection_time
     6    visit_num           Num     8    F8.             F8.            visit_num
     7    barcode_ID_9        Num     8    F8.             F8.            barcode_ID_9
     8    tube_type           Char    9    $11.            $11.           tube_type
     9    num_tubes           Num     8    F8.             F8.            num_tubes
```

CONCLUSIONS

This workshop has presented the basic processes for reading and writing XML data to and from SAS. SAS offers several ways to handle XML files depending on the structure of the XML data. If the data comes in a rectangular format, XML libname can process it directly. If the XML data has a complex hierarchical structure, you will need to create a map to inform SAS which elements you wish to extract and point to their location in the hierarchy of the source XML document. The new production-version XML Mapper tool, provided as a stand-alone application in SAS 9.1, greatly simplifies this process.

The files discussed in this paper will be available for download after the conference from the following URL:

http://www.mgcdata.com/presentations/XMLhandson910.zip

COPYRIGHT INFORMATION

REFERENCES

Castro, Elizabeth. XML for the World Wide Web. Peachpit Press: Berkeley, CA. 2001.

Friebel, Anthony. “XML? We do that!”. Paper 173-28. SUGI 28 Proceedings. SAS: Cary, NC. 2003.

SAS Institute Inc. 2003. SAS OnlineDoc® 9.1. SAS: Cary, NC. http://support.sas.com/91doc/docMainpage.jsp

“Using ODS to Export Markup Languages” SAS: Cary, NC 2002.
http://www.sas.com/rnd/base/topics/odsmarkup/index.html

CONTACT INFORMATION

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

Miriam Cisternas
Manager, Statistical Analysis
Ovation Research Group
5051 Millay Court
Carlsbad, CA 92008
(760) 804-1946
mcisternas@ovation.org

Ricardo Cisternas, Partner
MGC Data Services
ricardo@mgcdata.com
www.mgcdata.com