Paper 030-2010

Using XML Mapper and Enterprise Guide to Read Data and Metadata from an XML File

Larry Hoyle, Institute for Policy & Social Research, Univ. of Kansas, Lawrence, KS

ABSTRACT

Increasingly, data and metadata are archived together in XML files. This tutorial works through the process of reading an XML file containing a data table, variable labels, notes about variables, and multiple sets of labels for values – some of which are not associated with any variable.
First, we will use the XML Mapper with the archetypical XML file to create an XMLMAP file. We will then create an Enterprise Guide project to organize the sequence of operations that follow. The SAS XMLMAP facility of the XML engine uses the XMLMAP file to read the complex, hierarchical, XML file into a set of related tables. Additional SAS code transforms the initial data table from one row per value to a table with multiple variables and one row per observation. Other code labels the dataset, applies variable labels, creates a CNTLIN dataset, creates formats from the CNTLIN dataset, and associates formats with variables.
The finished Enterprise Guide project could then be used to read any XML file having the same structure (Schema) as the original file.

INTRODUCTION

Approaches to representing datasets in XML can be grouped into two different styles. In the first style, element or attribute names carry content themselves. The example XML to the right shows the SAS dataset "MYSAS1" output with the XML libname engine using the xmltype=generic option. Each row is contained in an element named for the dataset, in this instance "<MYSAS1>". Here each dataset has variables "n", "charvar" and "created". The xmltypes "ORACLE" and "MSACCESS" share this approach. The XML is fairly human-readable, but only tables with exactly the same list of variables in the same order would have the same structure. In order to be validated, each different structure would require its own XML schema. The SAS XML engine can import XML files with this type of simple structure without any additional instructions.

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
  <MYSAS1>
    <n>1</n>
    <charvar>one</charvar>
    <created>1579430009.854</created>
  </MYSAS1>
  <MYSAS1>
    <n>2</n>
    <charvar>two</charvar>
    <created>1579430009.854</created>
  </MYSAS1>
 </TABLE>
```

The second style of XML uses consistent element and attribute names for any dataset. In this piece of an XML file, exported using "xmltype=export", rows are contained in an element "<DATA>" and columns are in "<DATUM-NUMERIC >" or "<DATUM>" elements, with the variable name contained in the "name" attribute of the element. This style is somewhat less "human-readable" and a bit more verbose, but is much more machine-actionable, in that a program always knows to look for rows in "<DATA>" elements and columns in "<DATUM…" elements. For the SAS libname engine the instructions for where to look for values are contained in an **XMLMap** file.

```
 <TABLE-DATA>
  <DATA>
    <DATUM-NUMERIC name="n">10</DATUM-NUMERIC>
    <DATUM name="charvar">ten</DATUM>
    <DATUM-NUMERIC name="created">1579430010.197</DATUM-NUMERIC>
  </DATA>
  <DATA>
    <DATUM-NUMERIC name="n">11</DATUM-NUMERIC>
    <DATUM name="charvar">eleven</DATUM>
    <DATUM-NUMERIC name="created">1579430010.197</DATUM-NUMERIC>
  </DATA>
</TABLE-DATA>
```

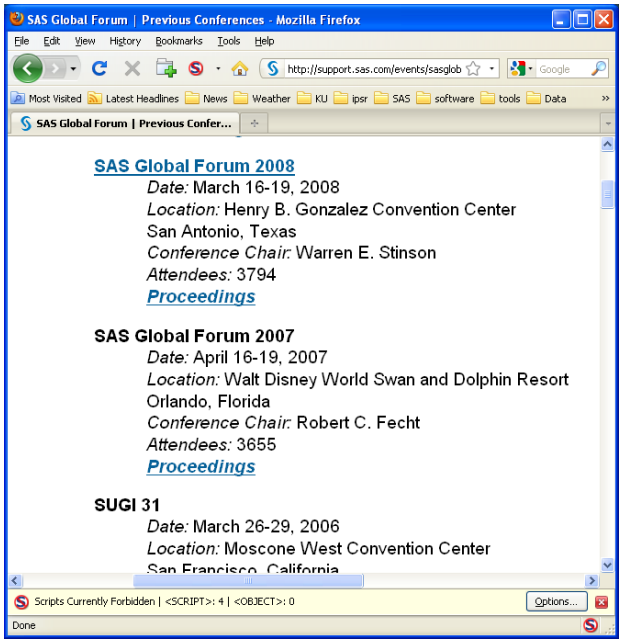
An XML file may contain not only data but also metadata. Here is another piece of the XML file exported using "xmltype=export". This "<COLUMN>" element contains metadata for the column named "created". It is the third variable. It is numeric, represented in the XML as a floating point number, and it has the SAS format of "B8601DT". Note that in the first example these metadata were lost. The SAS xmltype export also may contain metadata about the original library and dataset such as SAS version and creation date.

```
    <TABLE-METADATA>
…
      <COLUMN order="3" name="created">
        <TYPE>numeric</TYPE>
        <DATATYPE>float</DATATYPE>
        <FORMAT>B8601DT</FORMAT>
        <Offset>18</Offset>
    </TABLE-METADATA>
```

An XMLMap file is an XML file which contains a description of how to translate (or "map" in the mathematical sense) from the hierarchical XML file into the relational model that can be represented in SAS datasets. One XMLMap file can be used to import any XML file having the same structure. The XMLMap file is just text and can be created with any text editor, but there is an easier way to create an XMLMap file. SAS provides a program, the "XML Mapper" which allows a user to create an XMLMap file by dragging and dropping elements from a representation of an XML structure. XML Mapper can determine this structure from an example XML file or, if one exists, from an XML Schema – a formal description of the XML structure.

OUR EXAMPLE

This paper will show the process of designing a procedure to import any XML file with the same structure as our example XML file. The particular data we use doesn't matter as long as we have all of the structural elements we might encounter in any future dataset. Our initial data will be taken from the SGF Web site listing of previous SGF/SUGI conferences. (For the complete XML data, see Appendix 4.)



Original Source

XML with <descriptors> and <expansion> collapsed

The data were entered into Microsoft Excel and the file looks like this:

name	confurl	location	city	state	chair	attendees	proceedingsurl	startdate	enddate	OnLexJansen	season
SUGI 31		Moscone Wes	San Francisco	California	Kim LeBouton	4242	http://www2.sas.com/procee	26-Mar-06	29-Mar-06	Complete	Spring
SAS Global Forum 2007		Walt Disney W	Orlando	Florida	Robert C. Fecht	3655	http://www2.sas.com/procee	16-Apr-07	19-Apr-07	Complete	Spring

In order to make this a real-world example, the Excel file was then imported into Stata where some additional metadata were added and then the data were exported into a Stata "dta" format XML file. In this structure, rows of the data are contained in "<o>" elements and columns are contained in "<v>" elements. Within each "<o>" element columns are identified only by their order. Missing values are represented by an empty "<v>" element.

Names of the variables, in order, are stored in separate "<variable>" elements inside a "<varlist>" element.

```
<varlist>
<variable varname='name'/>
<variable varname='confurl'/>
<variable varname='location'/>
<variable varname='city'/>
<variable varname='state'/>
<variable varname='chair'/>
<variable varname='attendees'/>
<variable varname='proceedingsurl'/>
<variable varname='startdate'/>
<variable varname='enddate'/>
<variable varname='OnLexJansen'/>
<variable varname='season'/>
</varlist>
```

```
<o>
<v>SUGI 31</v>
<v></v>
<v>Moscone West Convention Center</v>
<v>San Francisco</v>
<v>California</v>
<v>Kim LeBouton</v>
<v>4242</v>
<v>http://www2.sas.com/proceedings/sugi31/toc.html</v>
<v>16886.000000000000000</v>
<v>16889.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
```

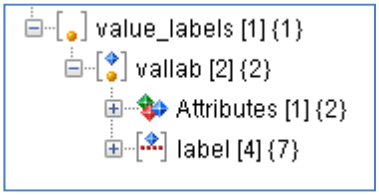
The XML file also contains metadata. Value labels, for example are contained in a “<value_labels>” element, with each set of labels appearing in a “<vallab>” element having a “name” attribute and a sequence of “label” elements containing value and label pairs. The “<value_labels>” element conveniently corresponds to a SAS PROC FORMAT.

```
<value_labels>
  <vallab name='Season'>
    <label value='1'>Winter</label>
    <label value='2'>Spring</label>
    <label value='3'>Summer</label>
    <label value='4'>Fall</label>
  </vallab>
  <vallab name='OnJansen'>
    <label value='0'>None</label>
    <label value='1'>Partial</label>
    <label value='2'>Complete</label>
  </vallab>
</value_labels>
```

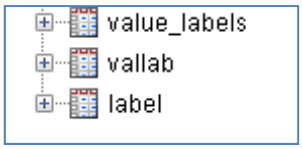
```
Proc format;
  Value Season
     1 = "Winter"
     2 = "Spring"
     3 = "Summer"
     4 = "Fall"          ;
  Value OnJansen
     0 = "None"
     1 = "Parital"
     2 = "Complete"    ;
Run;
```

PARSING THE XML FILE WITH XML MAPPER

The first challenge in using our sample XML file is to convert from the somewhat complex hierarchical structure to a set of relational tables. The XML libname engine allows for a reference to the XMLMap file that specifies the desired translation from the hierarchy to the relational structure. The XML Mapper program is a user friendly tool for building that mapping. The “<value_labels>” element above, for example, can be represented in three tables, one for each level of the hierarchy. Since each “<value_labels>” element might contain a variable number of “<vallab>” elements, a “value_labels” table can have a a 1 to many relationship with a “vallab” table which, in turn, can have a 1 to many relationship with a “label” table. (The complete XML Map file for this project can be found in Appendix 5.)

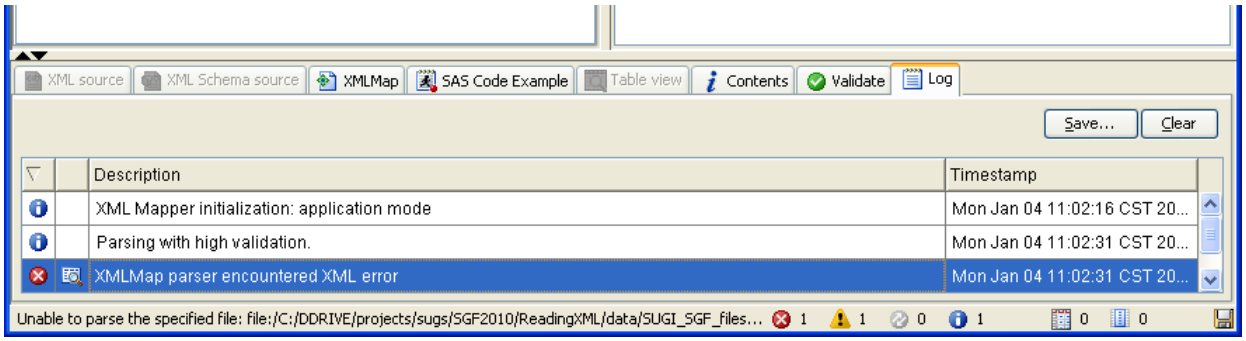


XML Structure

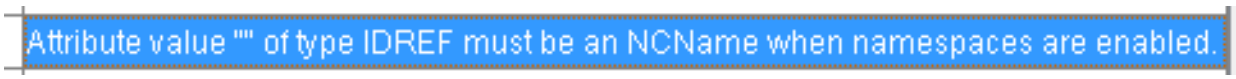


Tables

Opening our archetypal XML file in XML Mapper reveals a problem. The XML file contains an embedded document type definition (DTD), which should describe the structure of a valid XML file of this document type. Unfortunately, the document does not quite match the description of the structure and by default XML Mapper refuses to open it.

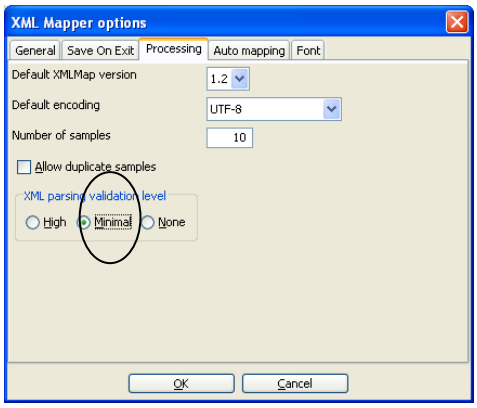


Double clicking on “XMLMAP parser encountered XML error” reveals the problem through the somewhat cryptic message:

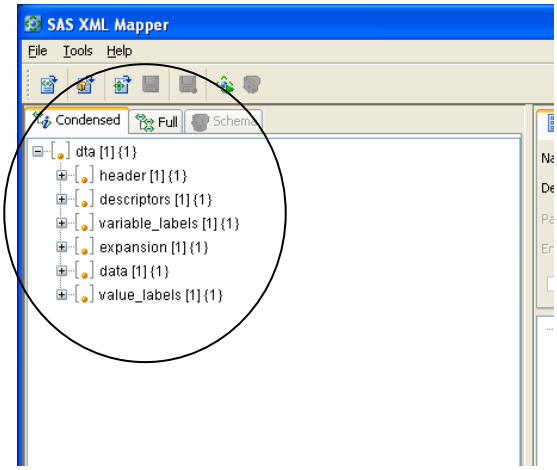


The problem turns out to be the empty values in the varname attributes of some “<sort>” elements. We can try to work around this problem by setting parsing validation level to minimal in “Tools…Options…Processing”.

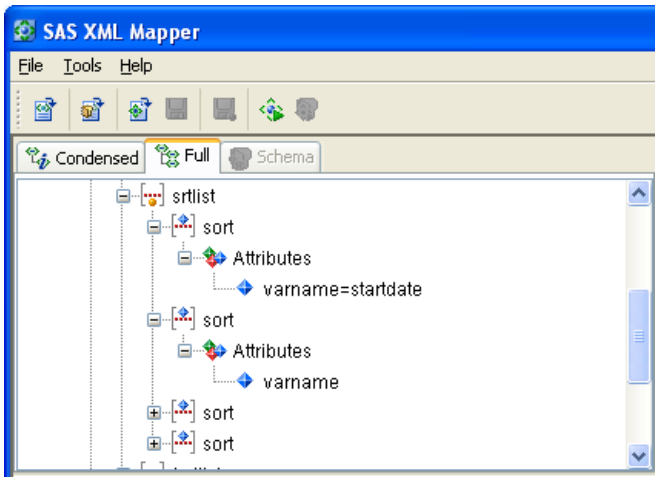
```
<srtlist>
    <sort varname='startdate'/>
    <sort varname=''/>
    <sort varname=''/>
    <sort varname=''/>
</srtlist>
```



Closing XML Mapper and reopening the XML file succeeds and XML Mapper displays an outline of the structure of our file.

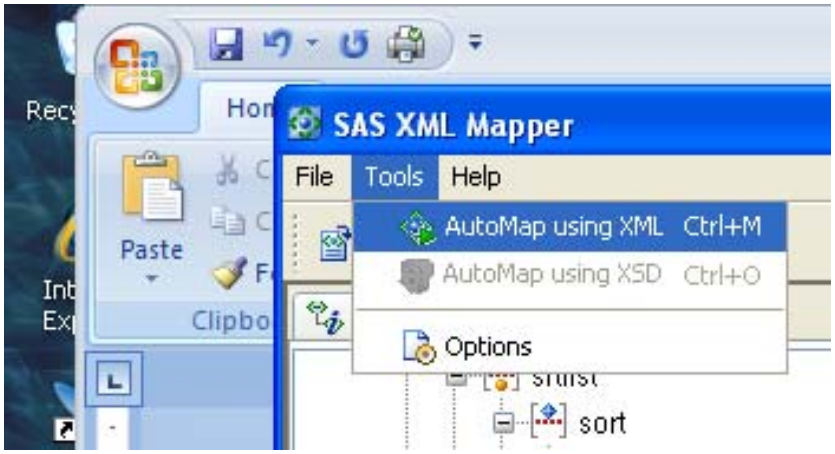


Viewing the XML shows that the <srtlist> element looks ok.



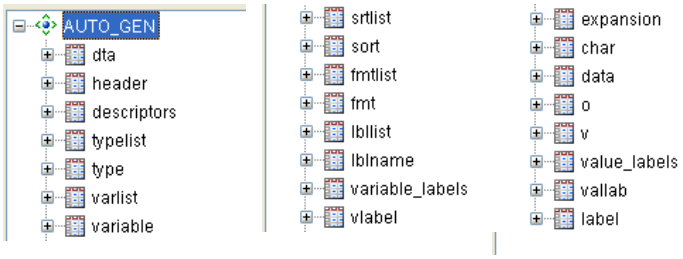
BUILDING THE XML MAP

In XML Mapper, we could explicitly drag and drop elements and attributes to create the tables we want, but the AutoMap facility under “Tools….AutoMap using XML” will create a complete relational structure automatically.



XML Mapper creates an XMLMap file and sample SAS code that reads the XML file and produces SAS datasets. We can use the SAS code and XMLMap file in what follows.

The Automap facility generates these tables:



The “<value_labels>” element is mapped into three tables. Note the “_ORDINAL” variables. These are generated automatically by the Automap tool. The value of “1” for dta_ORDINAL in the **value_labels** table indicates that that row links to the first, and in this case only, row in the **dta** table. (The “<dta>” element is the root element in the XML file.) Similarly, value_labels_ORDINAL links records in the **vallab** table to records in the **value_labels table**. Neither of these links will be of much use to us since the “<dta>” element and the “<value_labels>” element are containers that just occur once in the XML file.

What will be of use is that values for vallab_ORDINAL link the **label** table to the **vallab** table. The figure below highlights the records with vallab_ORDINAL of 2, which are a set of value labels with the name of “Season”. Those value-label combinations are shown below expressed in a PROC FORMAT statement.

```
<value_labels>
<vallab name='SeasonFR'>
<label value='1'>Hiver</label>
<label value='2'>Printemps</label>
<label value='3'>Ete</label>
<label value='4'>Automne</label>
</vallab>
<vallab name='Season'>
<label value='1'>Winter</label>
<label value='2'>Spring</label>
<label value='3'>Summer</label>
<label value='4'>Fall</label>
</vallab>
<vallab name='OnJansen'>
<label value='0'>None</label>
<label value='1'>Partial</label>
<label value='2'>Complete</label>
</vallab>
</value_labels>
```

value_labels XML element

```
Proc format;
  Value Season
     1 = "Winter"
     2 = "Spring"
     3 = "Summer"
     4 = "Fall"          ;
  Value OnJansen
     0 = "None"
     1 = "Parital"
     2 = "Complete"    ;
```

PROC FORMAT from <value_labels>

	dta_ORDINAL
1	1

dta table

	dta_ORDINAL	value_labels_ORDINAL
1	1	1

value_labels table

	value_labels_ORDINAL	vallab_ORDINAL	name
1	1	1	SeasonFR
2	1	2	Season
3	1	3	OnJansen

vallab table

	vallab_ORDINAL	label_ORDINAL	value	label
1	1	1	1	Hiver
2	1	2	2	Printemps
3	1	3	3	Ete
4	1	4	4	Automne
5	2	5	1	Winter
6	2	6	2	Spring
7	2	7	3	Summer
8	2	8	4	Fall
9	3	9	0	None
10	3	10	1	Partial
11	3	11	2	Complete

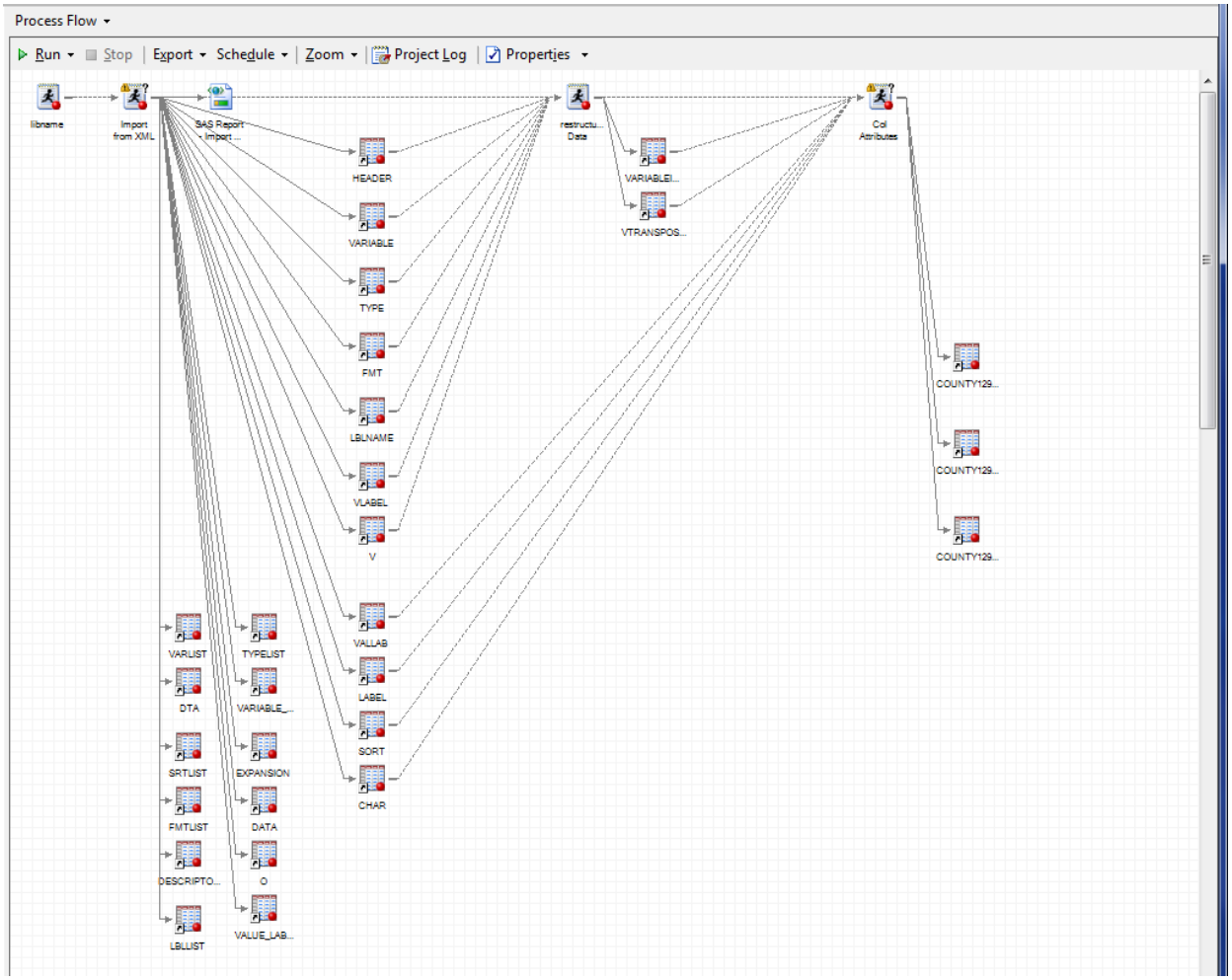
label table

WORKING WITH THE SET OF TABLES

The next step in this project is to transform the data in the multiple tables into the single dataset that they represent. This will involve several smaller steps

- Transposing the data table into one row per observation from the one row per number layout
- Converting from text to numeric for the numeric columns
- Building and running a CNTLIN dataset to create formats
- Renaming and applying attributes to columns, including formats
- Preserving any other metadata that cannot be represented in the single SAS dataset

Since this clearly involves several steps operating on different subsets of the tables, Enterprise Guide (EG) offers the capability of documenting this process as a process flow. The final basic process flow diagram is shown below. It shows a sequence of code nodes across the top and shows which tables are inputs and outputs from each step in the process as well as which steps are prerequisites for other steps. EG also has the capability of accepting the XML file name as a parameter and turning this into a stored process.

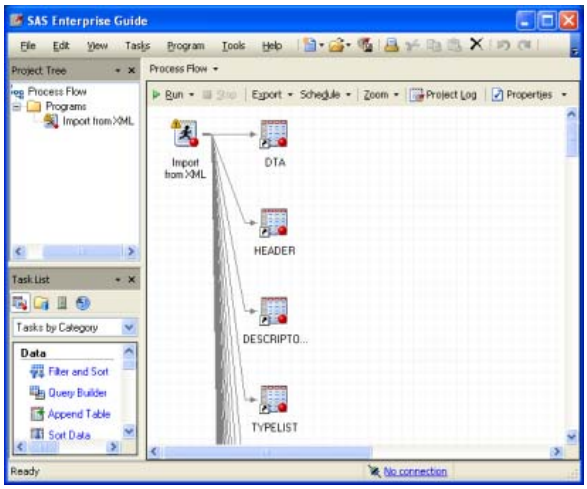


Importing the SAS code generated by XML Mapper into Enterprise Guide and running it shows the 23 tables created.

This code generates the following warning flag:

"WARNING: DOCTYPE element encountered. The SAS XML Libname Engine does not support processing of Data Type Definitions (DTD). External entity references in the document will not be resolved, and no mark up validation will be performed."

In our case, it is not a problem that the DTD will be ignored, but it was good to know.



The two figures below show a segment of the SAS code generated by XML Mapper and the resultant output.

```
filename  SUGISGF3 'Q:\SUGI_SGF4.xml';
filename  SXLEMAP 'Q:\SGF_030_2010_XMLmap.map';
libname  SUGISGF3 xml xmlmap=SXLEMAP access=READONLY;

title 'Table vallab';
proc print data=SUGISGF3.vallab(obs=30); run;
title;

title 'Table label';
proc print data=SUGISGF3.label(obs=30); run;
title;
```

Table vallab

Obs	value_labels_ORDINAL	vallab_ORDINAL	name
1	1	1	SeasonFR
2	1	2	Season
3	1	3	OnJansen

Page Break

Table label

Obs	vallab_ORDINAL	label_ORDINAL	value	label
1	1	1	1	Hiver
2	1	2	2	Printemp
3	1	3	3	Ete
4	1	4	4	Automne
5	2	5	1	Winter
6	2	6	2	Spring
7	2	7	3	Summer
8	2	8	4	Fall
9	3	9	0	None
10	3	10	1	Partial
11	3	11	2	Complete

The EG results tab shows PROC CONTENTS and PROC PRINT on each of the tables. Inspection of these tables reveals which of them will be useful in the remainder of the project.
The tables mapped from the XML structure provide the following types of data or metadata. Eleven of the tables represent higher level, single occurrence containers that can be ignored for this project.

Content / Role	Table(s)
Data - columns	**v**
Data - rows	**o**
List of variables	**variable**
Variable labels	**vlabel**
Data type and length	**fmt, type**
Dataset information - label, timestamp, nobs etc	**header, char**
Sort order	**sort**
Value labels (formats)	**vallab, label**
Link formats to variables	**lblname**
Other variable characteristics	**char**
Unused tables	**dta, varlist, srtlist, fmtlist, descriptors, lbllist, typelist, variable_labels, expansion, data, value_labels**

RESTRUCTURE THE DATA FILE

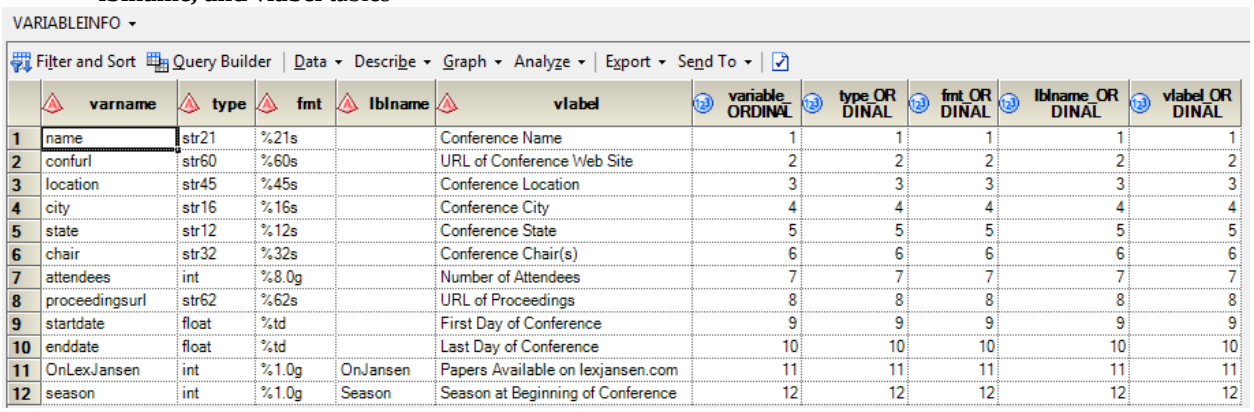
The first program node in this project was the SAS code imported from XML Mapper (see Appendix 1) that produced the 23 raw tables. The next node transforms those 23 tables into three tables (see Appendix 2).

- A simple PROC TRANSPOSE, **vtransposed**, of the table **v** by o_ORDINAL where all the resultant data columns are character variables and variable names for the data are in the form COL<n> where n is the column number.


VTRANSPOSED

	o_ORDINAL	_NAME_	_LABEL_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9	COL10	COL11	COL12
1	1	v	v	SUGI '76		Hyatt World	Kissimmee	Florida	Julian Horwich	206		5869.000000000...	5871.000000000...	0	1
2	2	v	v	SUGI '77		Fairmont	New Orleans	Louisiana	Rod Helms	200		6212.000000000...	6214.000000000...	0	1
3	3	v	v	SUGI '78		Caesar's Palace	Las Vegas	Nevada	Mike Farrell and...	241		6604.000000000...	6606.000000000...	0	1
4	4	v	v	SUGI '79		Sheraton Sand K...	Clearwater	Florida	Ramon Littell an...	500		6968.000000000...	6970.000000000...	0	1
5	5	v	v	SUGI '80		St. Anthony Hotel	San Antonio	Texas	Rudolf Freund	600		7353.000000000...	7355.000000000...	0	1
6	6	v	v	SUGI '81		Dutch Inn	Lake Buena Vista	Florida	Kenneth L. Koon...	1020		7709.000000000...	7712.000000000...	0	1
7	7	v	v	SUGI '82		SF Hilton & Tower	San Francisco	California	Helene Cavior	1322		8080.000000000...	8083.000000000...	0	1
8	8	v	v	SUGI '83		New Orleans Ma...	New Orleans	Louisiana	J. Philip Miller	1656		8416.000000000...	8419.000000000...	0	1
9	9	v	v	SUGI '84		Diplomat Hotel	Hollywood Beach	Florida	Sally Carson	2092		8843.000000000...	8846.000000000...	1	1
10	10	v	v	SUGI 10		MGM Grand	Reno	Nevada	Mike Farrell and...	2500		9200.000000000...	9203.000000000...	0	1

- A single table, **variableinfo**, with attributes for each column, joined from the **variable, type, fmt, lblname, and vlabel** tables


VARIABLEINFO

	varname	type	fmt	lblname	vlabel	variable_ORDINAL	type_ORDINAL	fmt_ORDINAL	lblname_ORDINAL	vlabel_ORDINAL
1	name	str21	%21s		Conference Name	1	1	1	1	1
2	confurl	str60	%60s		URL of Conference Web Site	2	2	2	2	2
3	location	str45	%45s		Conference Location	3	3	3	3	3
4	city	str16	%16s		Conference City	4	4	4	4	4
5	state	str12	%12s		Conference State	5	5	5	5	5
6	chair	str32	%32s		Conference Chair(s)	6	6	6	6	6
7	attendees	int	%8.0g		Number of Attendees	7	7	7	7	7
8	proceedingsurl	str62	%62s		URL of Proceedings	8	8	8	8	8
9	startdate	float	%td		First Day of Conference	9	9	9	9	9
10	enddate	float	%td		Last Day of Conference	10	10	10	10	10
11	OnLexJansen	int	%1.0g	OnJansen	Papers Available on lexjansen.com	11	11	11	11	11
12	season	int	%1.0g	Season	Season at Beginning of Conference	12	12	12	12	12

- A CNTLIN table, **mydataset_cntlin**, formed by joining **vallab** and **label**, to be input to a PROC FORMAT to recreate the value labels.

MYDATASET_CNTLIN

Filter and Sort | Query Builder | Data | Describe

	fmtname	start	label
1	SeasonFR	1	Hiver
2	SeasonFR	2	Printemp
3	SeasonFR	3	Ete
4	SeasonFR	4	Automne
5	Season	1	Winter
6	Season	2	Spring
7	Season	3	Summer
8	Season	4	Fall
9	OnJansen	0	None
10	OnJansen	1	Partial
11	OnJansen	2	Complete

APPLYING ATTRIBUTES TO COLUMNS

The third program node takes the three tables created by the second node, along with the **sort** and **char** tables, and creates the final dataset and a dataset containing the other unused metadata. (See Appendix 3)
In order to perform this task, some metadata about the metadata were hard coded into the process. The *fmt* variable in the **variableinfo** table contains values like "%21s" or "%8.0g". These formats carry information about the variable type (the "s" indicates character, the "g" indicates numeric). These formats also indicate display widths. These "meta-metadata" were encoded into Perl regular expressions in an array (see ***array fmts*** in Appendix 3) which matched the category of *fmt* value with a SAS format along with the number of numeric components (width and/ or decimal places) of the format.
The Perl regular expression below matches values of *fmt* starting with a "%" or "%-" followed by two numbers separated by a decimal point, followed by a "gc" . The array matches values of *fmt* that conform to this pattern with the SAS format "COMMA" followed by an overall width and number of decimal places. The '2' indicates that there are both width and decimal place components to the format. Thus, if there were no user format assigned to the variable the process could assign a variable with an *fmt* of "%8.2gc" a SAS format of "COMMA8.2".

```
array fmts {&nPatterns,3} $ 30 _temporary_ (

'/^%-*(\d+)\.(\d+)gc/'  'COMMA'  '2'   /* gc general (with commas) maps to COMMA */
```

The resultant dataset is sorted in the order of the columns listed in the sort table.

MYDATASET

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	o_ORDINAL	name	confurl	location	city	state	chair	attendees	proceedingsurl	startdate	enddate	OnLexJansen	season
1	1	SUGI '76		Hyatt World	Kissimmee	Florida	Julian Horwich	206		26JAN76	28JAN76	None	Winter
2	2	SUGI '77		Fairmont	New Orleans	Louisiana	Rod Helms	200		03JAN77	05JAN77	None	Winter
3	3	SUGI '78		Caesar's Palace	Las Vegas	Nevada	Mike Farrell and...	241		30JAN78	01FEB78	None	Winter
4	4	SUGI '79		Sheraton Sand K...	Clearwater	Florida	Ramon Littell an...	500		29JAN79	31JAN79	None	Winter
5	5	SUGI '80		St. Anthony Hotel	San Antonio	Texas	Rudolf Freund	600		18FEB80	20FEB80	None	Winter
6	6	SUGI '81		Dutch Inn	Lake Buena Vista	Florida	Kenneth L. Koon...	1020		08FEB81	11FEB81	None	Winter
7	7	SUGI '82		SF Hilton & Tower	San Francisco	California	Helene Cavior	1322		14FEB82	17FEB82	None	Winter
8	8	SUGI '83		New Orleans Ma...	New Orleans	Louisiana	J. Philip Miller	1656		16JAN83	19JAN83	None	Winter
9	9	SUGI '84		Diplomat Hotel	Hollywood Beach	Florida	Sally Carson	2092		18MAR84	21MAR84	Partial	Winter
10	10	SUGI '10		MGM Grand	Reno	Nevada	Mike Farrell and	2500		10MAR85	13MAR85	None	Winter

The third program node also joins the original table **char** with the combined **variableinfo** table to create a dataset containing all of the metadata that can't be represented in a SAS dataset and associated user formats. This is done with an SQL query. Characteristics of the "_dta" object apply to the dataset as a whole.

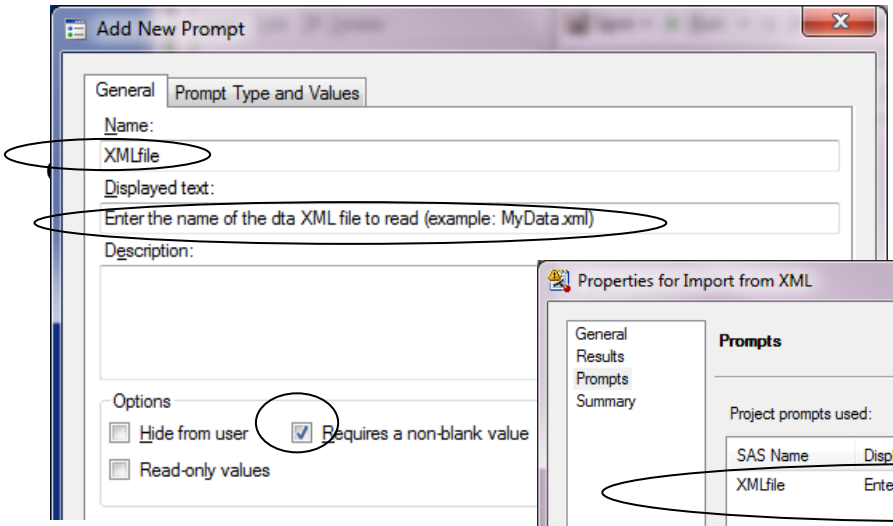
	object	objectLabel	characteristic Type	characteristic
1	OnLexJansen	Papers Available on lexjansen.com	note1	Indicates whether papers are available on Lex Jansen's SUGI paper site: http://www.lexjansen.com/sugi/
2	_dta	Dataset: MyDataset	_lang_c	default
3	_dta	Dataset: MyDataset	note1	"From 1976 Through 1984 the conferences were named SUGI 'YY where YY was the two digit year. From
4	_dta	Dataset: MyDataset	Source	http://support.sas.com/events/sasglobalforum/previous/index.html and http://www.lexjansen.com/sugi/
5	_dta	Dataset: MyDataset	Origin	http://support.sas.com/events/sasglobalforum/previous/index.html and http://www.lexjansen.com/sugi/
6	_dta	Dataset: MyDataset	_lang_list	default
7	chair	Conference Chair(s)	universe	Heroic, public spirited individuals
8	city	Conference City	universe	City with adequate facilities for SAS international conference
9	season	Season at Beginning of Conference	note1	can be labeled either by Season or SeasonFR
10	state	Conference State	universe	U.S. State or Canadian Province or Territory

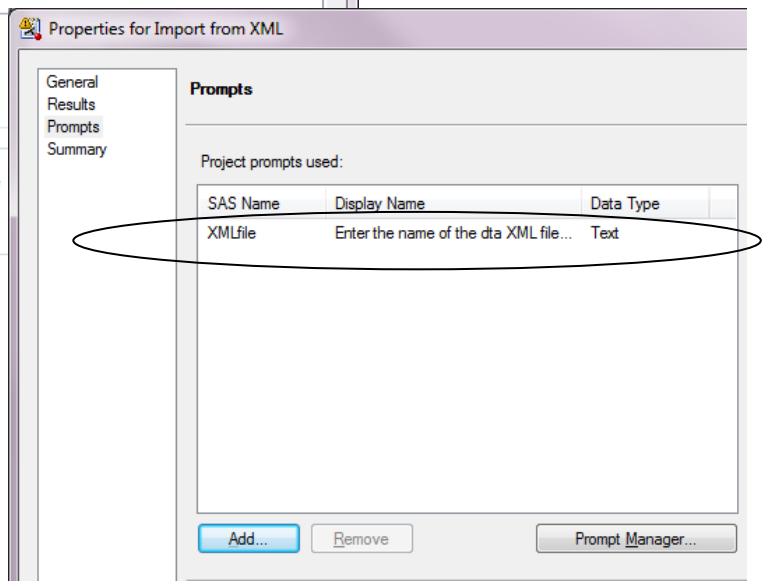
PARAMETERIZING THE PROCESS WITH PROMPTS

Enterprise Guide offers the capability to define "prompts". These in turn can be tied to task or program nodes so that when the node executes, EG prompts for entry of a value to be assigned to a macro variable. For this project, the code generated by XML Mapper can be edited to use a macro variable reference for the name of the XML file to be imported. Here the macro variable is named "XMLfile".

```
%let inFolder = C:\ ReadingXML\data\SUGI_SGF_files;
%let mapFolder = C:\ReadingXML\SAScode;
%let mapFile = SGF_030_2010_XMLmap.map;
filename  SUGISGF3 "&inFolder.\&XMLfile.";
filename  SXLEMAP "&mapFolder.\&mapFile.";
libname   SUGISGF3 xml xmlmap=SXLEMAP access=READONLY;
```

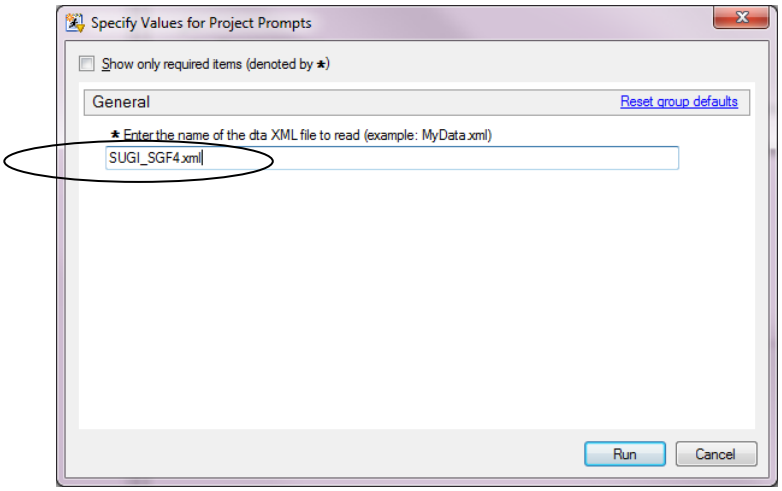
The next step is to create prompt using the Add function in the "Prompt Manager" pane.



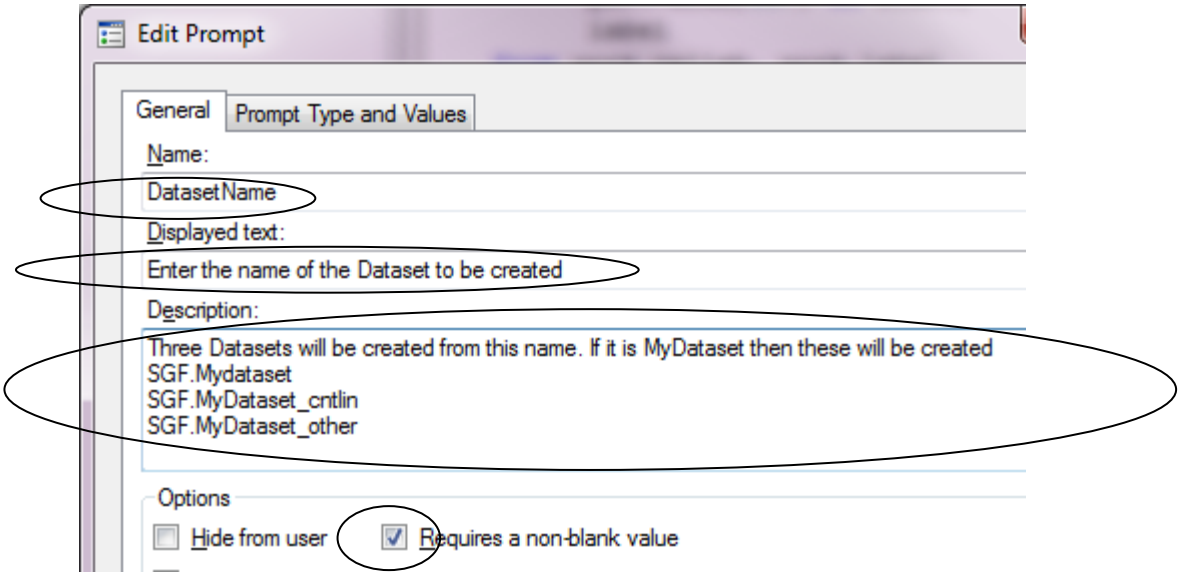


Then, through the properties of the node the prompt is assigned to the node.

When the node runs, this dialog box pops up to retrieve the value of the macro variable XMLfile.



The name of the output dataset can be inserted by defining a "DatasetName" prompt. The output dataset name will have a default value of "MyDataset". The names of the CNTLIN dataset and the dataset containing additional metadata will be formed from this name. Using the "Prompt type and Values" tab, a maximum length of 25 is assigned to this prompt so that the generated names do not exceed 32 characters in length (not shown).



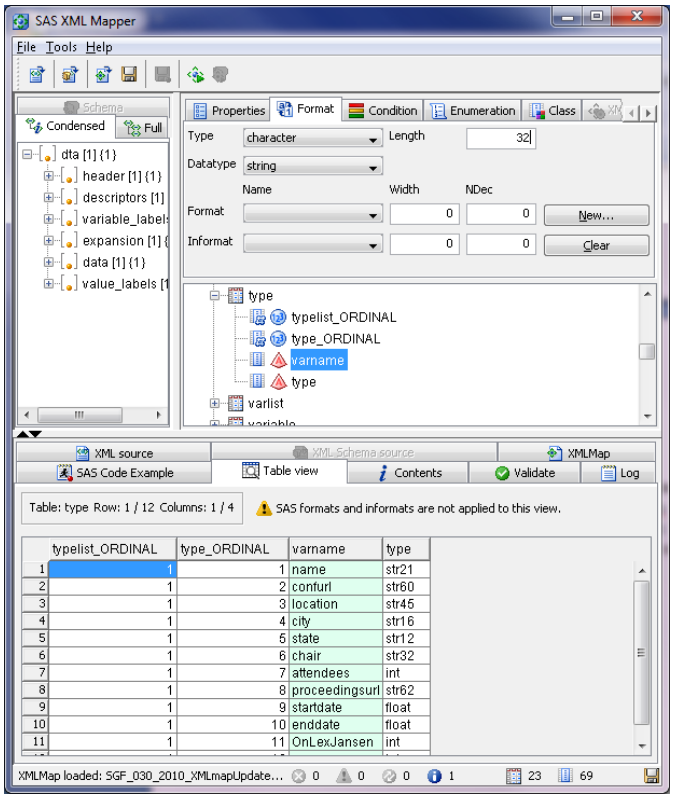
Here is the code that creates the CNTLIN dataset using the DatasetName entered by the user at run time.

```
proc SQL;
create table sgf.&DatasetName._CNTLIN as
 select name as fmtname,
        put(value,16.) as start,
        label
 from work.vallab, work.label
 where vallab.vallab_ORDINAL = label.vallab_ORDINAL;
quit;
```

TESTING

Testing the project with another dataset reveals a problem in using a representative file instead of a schema (which was not available in this case) to develop the XMLMap. The Automap procedure we used to create the XMLMap sets the length of the character fields to the maximum length found in the original XML file. This is a problem for things like variable and value labels, but is a really serious problem for variable names. Another input XML file may have variable names which are not unique when truncated to the length found in the first file. The solution lies in either making sure the representative file has at least one character string of each type that is the maximum length possible, or in using the properties tab of the column definition in the XML Mapper to set those lengths explicitly. The latter is quick and easy, a matter of looking at each of the table definitions and modifying the length property of the character fields.

In the example below the length of the variable name is set to 32 in the **type** table since we know that names in the "dta" XML cannot be longer than 32 characters. For this project all names are set to a maximum of 32 characters. Value labels in the dta file can be up to 32k long and notes can be up to 67,784. For this project both are set to a maximum of 2000 characters. Capturing notes of greater than 32K characters would require a different structure than the one used in this project due to the 32k limit on SAS character variable length.



CONCLUSION

Building this project was a lot of work for converting one file, but once built, the project can be used to import ANY XML file with the same structure. The Enterprise Guide process flow diagram provides a nice visual representation of the dependencies among project elements, as well as a convenient way to navigate among tables and code nodes and a front end for running the project.

This project also reveals an area in which the SAS system could be enhanced. Having to use three tables to save the dataset and associated metadata is workable, but not ideal. It would be nice to have at the least an XML export format that would allow a dataset and associated formats to be stored together. It would be even nicer to have an extensible format that would allow the storage of other metadata together with the data. This would allow for storage of metadata on the collection of the data (survey questions, sampling, universe, etc.), metadata useful for data discovery and attribution (author, title, sponsor, copyright provisions, etc.) and more all in one structure.

One approach to this might be an XML representation of something like RDF triples – (subject, verb, object) that could store information like (variable X, has a universe of, people from Europe).

RECOMMENDED READING

- **HOYLE, LARRY; WACKEROW, JOACHIM *EXPORTING SAS DATASETS TO DDI 3 XML FILES - DATA, METADATA, AND MORE METADATA*, PAPER 137-2008, SAS GLOBAL FORUM 2008, SAN ANTONIO, TEXAS, MARCH 2008.** http://www2.sas.com/proceedings/forum2008/137-2008.pdf

- **HOYLE, LARRY; READING MICROSOFT WORD XML FILES WITH SAS® SUGI31, SAS USER GROUP INTERNATIONAL CONFERENCE, SAN FRANCISCO, CA, 2006.** http://www2.sas.com/proceedings/sugi31/019-31.pdf

- **HOYLE, LARRY; USING XML MAPPER AND XMLMAP TO READ DATA DOCUMENTED BY DATA DOCUMENTATION INITIATIVE (DDI) FILES SUGI30 SAS USERS GROUP INTERNATIONAL CONFERENCE, PHILADELPHIA, PENNSYLVANIA, APRIL 2005.** http://www2.sas.com/proceedings/sugi30/099-30.pdf

- **JANSEN, LEX. USING THE SAS® XML MAPPER AND ODS PDF TO CREATE A PDF REPRESENTATION OF THE DEFINE.XML (THAT CAN BE PRINTED) NESUG 2008, PITTSBURGH, PA** http://www.nesug.org/Proceedings/nesug08/ph/ph06.pdf**.**

- **MARTELL, CAROL. SAS® *XML MAPPER TO THE RESCUE*. SAS GLOBAL FORUM PAPER 099-2008** http://www2.sas.com/proceedings/forum2008/099-2008.pdf**.**

- **PRATTER, FREDERICK. *XML FOR SAS® PROGRAMMERS*. SAS GLOBAL FORUM PAPER 042-2008** http://www2.sas.com/proceedings/forum2008/042-2008.pdf**.**

- **SAS INSTITUTE INC. 2009. *SAS® 9.2 XML LIBNAME ENGINE: USER'S GUIDE*. CARY, NC: SAS INSTITUTE INC.** **HTTP://SUPPORT.SAS.COM/DOCUMENTATION/CDL/EN/ENGXML/61740/PDF/DEFAULT/ENGXML.PDF**

- **W3C. *RDF SEMANTIC WEB STANDARDS*.** http://www.w3.org/RDF/

CONTACT INFORMATION

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

Larry Hoyle
Institute for Policy & Social Research, University of Kansas
1541 Lilac Lane, Suite 607 Blake Hall
Lawrence, KS, 66045-3129
785 – 864 - 9110
LarryHoyle@ku.edu
http://www.ipsr.ku.edu/

NOTE: This paper, the associated Enterprise Guide Project, and the XML data files can be found at:

http://www.ipsr.ku.edu/ksdata/sashttp and on http://www.sascommunity.org

APPENDIX 1 – SAS CODE GENERATED BY XML MAPPER (“NODE 1”)

MODIFIED WITH MACRO VARIABLES

```
/*******************************************************************************
 *  Generated by XML Mapper, 902000.3.6.20090116170000_v920
 *******************************************************************************/
%let inFolder =
C:\DDRIVE\projects\sugs\SGF2010\ReadingXML\data\SUGI_SGF_files;

%let mapFolder = C:\DDRIVE\projects\sugs\SGF2010\ReadingXML\SAScode;

%let mapFile = SGF_030_2010_XMLmapUpdated.map;

filename  SUGISGF3 "&inFolder.\&XMLfile.";

filename  SXLEMAP "&mapFolder.\&mapFile.";

libname   SUGISGF3 xml xmlmap=SXLEMAP access=READONLY;


proc datasets lib=SUGISGF3; run;

/*
 *  Contents
 */

proc contents data=SUGISGF3.dta varnum; run;
proc contents data=SUGISGF3.header varnum; run;
proc contents data=SUGISGF3.descriptors varnum; run;
proc contents data=SUGISGF3.typelist varnum; run;
proc contents data=SUGISGF3.type varnum; run;
proc contents data=SUGISGF3.varlist varnum; run;
proc contents data=SUGISGF3.variable varnum; run;
proc contents data=SUGISGF3.srtlist varnum; run;
proc contents data=SUGISGF3.sort varnum; run;
proc contents data=SUGISGF3.fmtlist varnum; run;
proc contents data=SUGISGF3.fmt varnum; run;
proc contents data=SUGISGF3.lbllist varnum; run;
proc contents data=SUGISGF3.lblname varnum; run;
proc contents data=SUGISGF3.variable_labels varnum; run;
proc contents data=SUGISGF3.vlabel varnum; run;
proc contents data=SUGISGF3.expansion varnum; run;
proc contents data=SUGISGF3.char varnum; run;
proc contents data=SUGISGF3.data varnum; run;
proc contents data=SUGISGF3.o varnum; run;
proc contents data=SUGISGF3.v varnum; run;
proc contents data=SUGISGF3.value_labels varnum; run;
proc contents data=SUGISGF3.vallab varnum; run;
proc contents data=SUGISGF3.label varnum; run;

/*
 *  Printing
 */

title 'Table dta';  proc print data=SUGISGF3.dta(obs=30); run;  title;

title 'Table header';  proc print data=SUGISGF3.header(obs=30); run; title;

title 'Table descriptors';  proc print data=SUGISGF3.descriptors(obs=30); run;  title;

title 'Table typelist';  proc print data=SUGISGF3.typelist(obs=30); run;  title;

title 'Table type';  proc print data=SUGISGF3.type(obs=30); run;  title;

title 'Table varlist';  proc print data=SUGISGF3.varlist(obs=30); run;  title;

title 'Table variable';  proc print data=SUGISGF3.variable(obs=30); run;  title;
```

```
title 'Table srtlist';  proc print data=SUGISGF3.srtlist(obs=30); run;  title;

title 'Table sort';  proc print data=SUGISGF3.sort(obs=30); run;  title;

title 'Table fmtlist';  proc print data=SUGISGF3.fmtlist(obs=30); run;  title;

title 'Table fmt';  proc print data=SUGISGF3.fmt(obs=30); run;  title;

title 'Table lbllist';  proc print data=SUGISGF3.lbllist(obs=30); run;  title;

title 'Table lblname';  proc print data=SUGISGF3.lblname(obs=30); run;  title;

title 'Table variable_labels';  proc print data=SUGISGF3.variable_labels(obs=30); run;  title;

title 'Table vlabel';  proc print data=SUGISGF3.vlabel(obs=30); run;  title;

title 'Table expansion';  proc print data=SUGISGF3.expansion(obs=30); run;  title;

title 'Table char';  proc print data=SUGISGF3.char(obs=30); run;  title;

title 'Table data';  proc print data=SUGISGF3.data(obs=30); run;  title;

title 'Table o';  proc print data=SUGISGF3.o(obs=30); run;  title;

title 'Table v';  proc print data=SUGISGF3.v(obs=30); run;  title;

title 'Table value_labels';  proc print data=SUGISGF3.value_labels(obs=30); run;  title;

title 'Table vallab';  proc print data=SUGISGF3.vallab(obs=30); run;  title;

title 'Table label';  proc print data=SUGISGF3.label(obs=30); run;  title;


/*
 *  Local Extraction
 */

DATA dta; SET SUGISGF3.dta; run;
DATA header; SET SUGISGF3.header; run;
DATA descriptors; SET SUGISGF3.descriptors; run;
DATA typelist; SET SUGISGF3.typelist; run;
DATA type; SET SUGISGF3.type; run;
DATA varlist; SET SUGISGF3.varlist; run;
DATA variable; SET SUGISGF3.variable; run;
DATA srtlist; SET SUGISGF3.srtlist; run;
DATA sort; SET SUGISGF3.sort; run;
DATA fmtlist; SET SUGISGF3.fmtlist; run;
DATA fmt; SET SUGISGF3.fmt; run;
DATA lbllist; SET SUGISGF3.lbllist; run;
DATA lblname; SET SUGISGF3.lblname; run;
DATA variable_labels; SET SUGISGF3.variable_labels; run;
DATA vlabel; SET SUGISGF3.vlabel; run;
DATA expansion; SET SUGISGF3.expansion; run;
DATA char; SET SUGISGF3.char; run;
DATA data; SET SUGISGF3.data; run;
DATA o; SET SUGISGF3.o; run;
DATA v; SET SUGISGF3.v; run;
DATA value_labels; SET SUGISGF3.value_labels; run;
DATA vallab; SET SUGISGF3.vallab; run;
DATA label; SET SUGISGF3.label; run;
```

APPENDIX 2 - RESTRUCTURE DATA PROGRAM NODE ("NODE 2")

```
/* restructure the data table - all character variables */
                         /* retrieve the number of variables as a macro variable  */
proc sql noprint;
select nvar into :nvars
from header;
                                      /* join all variable info  */
create table work.variableInfo as
select variable.varname,
       type.type,
       fmt.fmt,
       lblname.lblname,
       vlabel.vlabel,

       variable.variable_ORDINAL,
       type.type_ORDINAL,
       fmt.fmt_ORDINAL,
       lblname.lblname_ORDINAL,
       vlabel.vlabel_ORDINAL
from  variable,
       type,
       fmt,
       lblname,
       vlabel
where variable.varname=type.varname AND
       variable.varname=fmt.varname AND
       variable.varname=lblname.varname AND
       variable.varname=vlabel.varname
;

%put nvars=&nvars;
                                          /* create the transposed (all char vars)
dataset  */
proc transpose data=work.v
               out=work.vTransposed;
var v;
by o_ordinal;
run;
```

APPENDIX 3 – COLUMN ATTRIBUTES PROGRAM NODE ("NODE 3")

```
                                        /*  join proc format info */
proc SQL;
create table sgf.&DatasetName._CNTLIN as
 select name as fmtname,
       put(value,16.) as start,
          label
 from work.vallab, work.label
 where vallab.vallab_ORDINAL = label.vallab_ORDINAL;


quit;

/* run the cntlin */
proc format library=sgf cntlin=sgf.&DatasetName._CNTLIN;
run;



                        /* build an array of macro variables each containing */
                        /* an attribute and assignment statment              */
                        /* to create a named column from the corresponding col<n> */
data _null_;
                        /* %- indicates left justified - will be ignored */
%let nPatterns=20;
set work.variableInfo end=last;
length  SASformat $ 100 attribStmt assignStmt defStmt $ 2000;
                                       /* array of PRX patterns with corresponding */
                                        /*format root and number of */
                                        /*  numeric elements in the format */
 array fmts {&nPatterns,3} $ 30 _temporary_ (
     '/^%-*(\d+)\.(\d+)g/'   'BEST'   '2'      /* g general to BEST  */
     '/^%-*(\d+)\.(\d+)f/'   ' '      '2'      /* f fixed  */
     '/^%-*(\d+)\.(\d+)e/'   'E',     '2'      /* e exponential  E */
     '/^%(\d+)x/'            'HEX'    '2'      /* x hexadecimal HEX  */

     '/^%(\d+)H/'            'BINARY' '1'      /* H hilo (binary) BINARY  */
     '/^%(\d+)L/'            'BEST'   '1'      /* L lohi (binary) no SAS format */
                                               /*  use HEX  */

     '/^%-*(\d+)\.(\d+)gc/'  'COMMA'  '2'     /* gc general (with commas)          */
                                              /*    COMMA (would BEST be better    */
     '/^%-*(\d+)\.(\d+)fc/'  'COMMA'  '2'     /* fc fixed  (with commas) COMMA     */

     '/^%0(\d+)\.(\d+)f/'    'Z'      '2'     /* 0 f fixed (leading zeros) Z       */

     '/^%-*(\d+),(\d+)gc/'   'COMMAX' '2'     /* gc general (with commas as decimal */
                                              /*  separator) COMMAX (would BEST be  */
                                              /*  better                           */
     '/^%-*(\d+),(\d+)fc/'   'COMMAX' '2'     /* fc fixed  (with commas            */
                                              /*  as decimal separator) COMMAX     */

     '/^%t[cC]/'  'DATETIME'          '0'     /* tc or tC date/time DATETIME       */
                                              /*   base milliseconds from 1JAN1960  */

     '/^%td/'  'DATE'                 '0'     /* td date/time DATETIME             */
                                              /*   base days from 1JAN1960         */

     '/^%tw/'  'WEEKW'                '0'     /* tc or tC date/time WEEKW          */
                                              /*    base weeks from 1JAN1960       */
     '/^%tm/'  'YYMM'                 '0'     /* tc or tC date/time YYMM           */
                                              /*    base months from 1JAN1960      */
```

```
   '/^%tq/'  'YYQ'                  '0'     /* tc or tC date/time YYQ             */
                                            /*    base quarters from 1JAN1960     */
   '/^%th/'  'YYQR'                 '0'     /* tc or tC date/time use YYQR????    */
                                            /*    base half-years from 1JAN1960   */

   '/^%ty/'  'BEST'                 '0'     /* tc or tC date/time BEST            */
                                            /*    base years from 0               */
   '/^%tg/'  'BEST'                 '0'     /* tc or tC date/time BEST            */
                                            /     base generic from 1JAN1960      */

                                            /* %- indicates left justified -      */
                                            /*   will be ignored                  */
                                            /* %~ indicates centered - ignored    */
   '/^%[-~]*(\d+)s/'  '$'           '1'     /* s string $   */

  );

array prxIDs {&nPatterns} _TEMPORARY_;

if _n_=1 then do ixPattern = 1 to &nPatterns;
  prxIDs{ixPattern} = prxParse(fmts{ixPattern,1});
end;

do ixPattern = 1 to &nPatterns;
  if prxmatch(prxIDs{ixPattern},fmt) then do;
    select ( fmts{ixPattern,3} );                 /* how many numeric components */
        when ('2')  SASformat = cats(fmts{ixPattern,2},
                                     prxposn(prxIDs{ixPattern}, 1, fmt),
                                     '.',
                                     prxposn(prxIDs{ixPattern}, 2, fmt),
                                     '.'
                                     );
        when ('1')  SASformat = cats(fmts{ixPattern,2},
                                     prxposn(prxIDs{ixPattern}, 1, fmt),
                                     '.'
                                     );

       otherwise     SASformat = cats(fmts{ixPattern,2},
                                      '.'
                                      );
    end; /* select */
                          /*  If BEST is the preferred format just use length 16  */
    if fmts{ixPattern,2} = 'BEST' then SASformat = 'BEST16.';

    if fmts{ixPattern,2} = '$' then do;

      SASlength = catx(' ',  '$',  prxposn(prxIDs{ixPattern}, 1, fmt)  );
    end;
    else do;
      SASlength = '8';
    end;

  end; /* if prxmatch */
end; /*  do ixPattern */

if lblname ne ' ' then SASformat = cats(lblname,'.');
                                                /* clean varname and vlabel */
                                                /* build macro variable content  */
                                                /* ATTRIBUTES */
attribStmt = catx(' ',
                  'ATTRIB',
                  varname,
                  'FORMAT',
```

```
                    '=',
                    SASformat,
                    'LENGTH',
                    '=',
                    SASlength,
                    'LABEL',
                    '=',
                    cats("'",  vlabel,  "'"),
                    ';'
                    );
                             /* ASSIGNMENT statement */
                             /* NOTE some date/time constructs need to be rescaled */


  if  SASlength = '8' then do;    /* numeric variable */
    assignStmt = catx(' ',
                      varname,
                      cats('= put(COL',
                           put(variable_ORDINAL,16.),
                           ',16.)'
                          )   ,
                      ';'
                     );
  end;
  else do;                        /*  string variable */
    assignStmt = catx(' ',
                      varname,
                      cats('= COL',
                           put(variable_ORDINAL,16.)
                           )   ,
                      ';'
                     );
  end;


                                        /*  assign a macro variable "array" element */

 call symput(cats('C_' , put(variable_ORDINAL,7.)  ) ,
             catx(' ',  attribStmt , assignStmt)
            );

 if last then call symput('NVARS',strip(put(variable_ORDINAL,7.) ) );

 run;


               /*  Use the macro vars to rename variables with attributes applied  */
 %macro restructure;
data sgf.&DatasetName.;
  set work.vtransposed ;
  drop _NAME_ _LABEL_ COL1-COL&NVARS;

  %DO IXV = 1 %TO &NVARS;
   &&C_&IXV;

  %END;
 run;
%mend restructure;

%restructure;
```

```
                /*  sort the dataset as specified in work.sort  */

proc sql noprint;
 select varname into :sortBy separated by ' '
 from work.sort
 where varname ne ' ';
  quit;
%put sortBy = &sortBy;


                      /*  get the first sort variable name */
proc sql noprint;
 select varname into :FirstSortVar
 from work.sort
 where sort_ORDINAL=1;
 quit;
 %put FirstSortVar = &FirstSortVar;


                      /* sort the file if there is at least one sort variable */
%macro SortFile();
%IF &FirstSortVar NE   %THEN %DO;
proc sort data=sgf.&DatasetName.;
  by &sortBy;
run;
%END;
%mend SortFile;
%sortFile();

                                     /* make a table with the metadata that */
                                     /* can't be stored in the SAs dataset  */
proc sql;
 create table sgf.&DatasetName._Other as
   select char.vname as object,
          case
           when char.vname = '_dta' then "Dataset: &DatasetName."
           else variableInfo.vlabel
          end as objectLabel,
          char.name as characteristicType,
          char.char as characteristic

   from work.char left join work.variableInfo on
     char.vname = variableInfo.varname
   where      char.char ne '1'
   order by object;
quit;
```

APPENDIX 4 THE “DTA” XML FILE

```
<?xml version="1.0" encoding="US-ASCII" standalone="yes"?>
<!DOCTYPE dta [
<!ELEMENT dta (header, descriptors,
variable_labels, expansion,
data, value_labels)>
<!ELEMENT header (ds_format, byteorder,
filetype, nvar, nobs, data_label,
time_stamp)>
<!ELEMENT ds_format (#PCDATA)>
<!ELEMENT byteorder (#PCDATA)>
<!ELEMENT filetype (#PCDATA)>
<!ELEMENT nvar (#PCDATA)>
<!ELEMENT nobs (#PCDATA)>
<!ELEMENT data_label (#PCDATA)>
<!ELEMENT time_stamp (#PCDATA)>
<!ELEMENT descriptors (typelist, varlist,
srtlist, fmtlist, lbllist)>
<!ELEMENT typelist (type)+>
<!ELEMENT type (#PCDATA)>
<!ATTLIST type varname IDREF #REQUIRED>
<!ELEMENT varlist (variable)+>
<!ELEMENT variable EMPTY>
<!ATTLIST variable varname ID #REQUIRED>
<!ELEMENT srtlist (sort)*>
<!ELEMENT sort EMPTY>
<!ATTLIST sort varname IDREF #REQUIRED>
<!ELEMENT fmtlist (fmt)+>
<!ELEMENT fmt (#PCDATA)>
<!ATTLIST fmt varname IDREF #REQUIRED>
<!ELEMENT lbllist (lblname)*>
<!ELEMENT lblname (#PCDATA)>
<!ATTLIST lblname varname IDREF #REQUIRED>
<!ELEMENT variable_labels (vlabel)*>
<!ELEMENT vlabel (#PCDATA)>
<!ATTLIST vlabel varname IDREF #REQUIRED>
<!ELEMENT expansion (char)*>
<!ELEMENT char (#PCDATA)>
<!ATTLIST char name  NMTOKEN #REQUIRED>
<!ATTLIST char vname NMTOKEN #REQUIRED>
<!ELEMENT data (o)*>
<!ELEMENT o (v)*>
<!ATTLIST o num NMTOKEN #IMPLIED>
<!ELEMENT v (#PCDATA)*>
<!ATTLIST v varname IDREF #IMPLIED>
<!ELEMENT value_labels (vallab)*>
<!ELEMENT vallab (label)+>
<!ATTLIST vallab name  NMTOKEN #REQUIRED>
<!ELEMENT label (#PCDATA)>
<!ATTLIST label value NMTOKEN #REQUIRED>
]>
<dta>
<header>
<ds_format>113</ds_format>
<byteorder>LOHI</byteorder>
<filetype>1</filetype>
<nvar>12</nvar>
<nobs>34</nobs>
<data_label>Data on SAS International Conference</data_label>
<time_stamp>25 Jan 2010 15:01</time_stamp>
</header>
<descriptors>
<typelist>
<type varname='name'>str21</type>
<type varname='confurl'>str60</type>
<type varname='location'>str45</type>
<type varname='city'>str16</type>
<type varname='state'>str12</type>
<type varname='chair'>str32</type>
<type varname='attendees'>int</type>
<type varname='proceedingsurl'>str62</type>
<type varname='startdate'>float</type>
<type varname='enddate'>float</type>
<type varname='OnLexJansen'>int</type>
<type varname='season'>int</type>
</typelist>
<varlist>
<variable varname='name'/>
<variable varname='confurl'/>
<variable varname='location'/>
<variable varname='city'/>
<variable varname='state'/>
<variable varname='chair'/>
<variable varname='attendees'/>
<variable varname='proceedingsurl'/>
<variable varname='startdate'/>
<variable varname='enddate'/>
<variable varname='OnLexJansen'/>
<variable varname='season'/>
</varlist>
<srtlist>
<sort varname='startdate'/>
<sort varname=''/>
<sort varname=''/>
<sort varname=''/>
</srtlist>
<fmtlist>
<fmt varname='name'>%21s</fmt>
<fmt varname='confurl'>%60s</fmt>
<fmt varname='location'>%45s</fmt>
<fmt varname='city'>%16s</fmt>
<fmt varname='state'>%12s</fmt>
<fmt varname='chair'>%32s</fmt>
<fmt varname='attendees'>%8.0g</fmt>
<fmt varname='proceedingsurl'>%62s</fmt>
<fmt varname='startdate'>%td</fmt>
<fmt varname='enddate'>%td</fmt>
<fmt varname='OnLexJansen'>%1.0g</fmt>
<fmt varname='season'>%1.0g</fmt>
</fmtlist>
<lbllist>
<lblname varname='name'></lblname>
<lblname varname='confurl'></lblname>
<lblname varname='location'></lblname>
<lblname varname='city'></lblname>
<lblname varname='state'></lblname>
<lblname varname='chair'></lblname>
<lblname varname='attendees'></lblname>
<lblname varname='proceedingsurl'></lblname>
<lblname varname='startdate'></lblname>
<lblname varname='enddate'></lblname>
<lblname varname='OnLexJansen'>OnJansen</lblname>
<lblname varname='season'>Season</lblname>
</lbllist>
</descriptors>
<variable_labels>
<vlabel varname='name'>Conference Name</vlabel>
<vlabel varname='confurl'>URL of Conference Web Site</vlabel>
<vlabel varname='location'>Conference Location</vlabel>
<vlabel varname='city'>Conference City</vlabel>
<vlabel varname='state'>Conference State</vlabel>
<vlabel varname='chair'>Conference Chair(s)</vlabel>
<vlabel varname='attendees'>Number of Attendees</vlabel>
<vlabel varname='proceedingsurl'>URL of Proceedings</vlabel>
<vlabel varname='startdate'>First Day of Conference</vlabel>
<vlabel varname='enddate'>Last Day of Conference</vlabel>
```

<vlabel varname='OnLexJansen'>Papers Available on lexjansen.com</vlabel>
<vlabel varname='season'>Season at Beginning of Conference</vlabel>
</variable_labels>
<expansion>
<char name='note1' vname='season'>can be labeled either by Season or SeasonFR</char>
<char name='note0' vname='season'>1</char>
<char name='Source' vname='_dta'>http://support.sas.com/events/sasglobalforum/previous/index.html and http://www.lexjansen.com/sugi/</char>
<char name='Origin' vname='_dta'>http://support.sas.com/events/sasglobalforum/previous/index.html and http://www.lexjansen.com/sugi/</char>
<char name='universe' vname='chair'>Heroic, public spirited individuals</char>
<char name='universe' vname='city'>City with adequate facilities for SAS international conference</char>
<char name='universe' vname='state'>U.S. State or Canadian Province or Territory</char>
<char name='_lang_list' vname='_dta'>default</char>
<char name='_lang_c' vname='_dta'>default</char>
<char name='note1' vname='_dta'>"From 1976 Through 1984 the conferences were named SUGI 'YY where YY was the two digit year. From 1985 through 2006 the conferences were named SUGI nn, where nn was a sequential number beginnig with 10. From 2007 through the present the conferences have been named SGF YYYY, where YYYY is the 4 digit year."</char>
<char name='note0' vname='_dta'>1</char>
<char name='note1' vname='OnLexJansen'>Indicates whether papers are available on Lex Jansen's SUGI paper site: http://www.lexjansen.com/sugi/</char>
<char name='note0' vname='OnLexJansen'>1</char>
</expansion>
<data>
<o>
<v>SUGI '76</v>
<v></v>
<v>Hyatt World</v>
<v>Kissimmee</v>
<v>Florida</v>
<v>Julian Horwich</v>
<v>206</v>
<v></v>
<v>5869.000000000000000</v>
<v>5871.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '77</v>
<v></v>
<v>Fairmont</v>
<v>New Orleans</v>
<v>Louisiana</v>
<v>Rod Helms</v>
<v>200</v>
<v></v>
<v>6212.000000000000000</v>
<v>6214.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '78</v>
<v></v>
<v>Caesar's Palace</v>
<v>Las Vegas</v>
<v>Nevada</v>
<v>Mike Farrell and Rod Strand</v>
<v>241</v>
<v></v>
<v>6604.000000000000000</v>
<v>6606.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '79</v>
<v></v>
<v>Sheraton Sand Key</v>
<v>Clearwater</v>
<v>Florida</v>
<v>Ramon Littell and William Wilson</v>
<v>500</v>
<v></v>
<v>6968.000000000000000</v>
<v>6970.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '80</v>
<v></v>
<v>St. Anthony Hotel</v>
<v>San Antonio</v>
<v>Texas</v>
<v>Rudolf Freund</v>
<v>600</v>
<v></v>
<v>7353.000000000000000</v>
<v>7355.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '81</v>
<v></v>
<v>Dutch Inn</v>
<v>Lake Buena Vista</v>
<v>Florida</v>
<v>Kenneth L. Koonce</v>
<v>1020</v>
<v></v>
<v>7709.000000000000000</v>
<v>7712.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '82</v>
<v></v>
<v>SF Hilton & Tower</v>
<v>San Francisco</v>
<v>California</v>
<v>Helene Cavior</v>
<v>1322</v>
<v></v>
<v>8080.000000000000000</v>
<v>8083.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '83</v>
<v></v>
<v>New Orleans Marriott</v>
<v>New Orleans</v>
<v>Louisiana</v>
<v>J. Philip Miller</v>

<v>1656</v>
<v></v>
<v>8416.000000000000000</v>
<v>8419.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI '84</v>
<v></v>
<v>Diplomat Hotel</v>
<v>Hollywood Beach</v>
<v>Florida</v>
<v>Sally Carson</v>
<v>2092</v>
<v></v>
<v>8843.000000000000000</v>
<v>8846.000000000000000</v>
<v>1</v>
<v>1</v>
</o>
<o>
<v>SUGI 10</v>
<v></v>
<v>MGM Grand</v>
<v>Reno</v>
<v>Nevada</v>
<v>Mike Farrell and Rod Strand</v>
<v>2500</v>
<v></v>
<v>9200.000000000000000</v>
<v>9203.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI 11</v>
<v></v>
<v>Atlanta Hilton & Tower</v>
<v>Atlanta</v>
<v>Georgia</v>
<v>Don Henderson</v>
<v>2600</v>
<v></v>
<v>9536.000000000000000</v>
<v>9539.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI 12</v>
<v></v>
<v>Loews Anatole</v>
<v>Dallas</v>
<v>Texas</v>
<v>Pat Hermes Smith</v>
<v>2800</v>
<v></v>
<v>9900.000000000000000</v>
<v>9903.000000000000000</v>
<v>1</v>
<v>1</v>
</o>
<o>
<v>SUGI 13</v>
<v></v>
<v>Orlando World Marriott</v>
<v>Orlando</v>
<v>Florida</v>
<v>Robert M. Hamer and Gerald Hobbs</v>
<v>3500</v>
<v></v>
<v>10313.000000000000000</v>
<v>10316.000000000000000</v>
<v>1</v>
<v>2</v>
</o>
<o>
<v>SUGI 14</v>
<v></v>
<v>SF Hilton & Tower</v>
<v>San Francisco</v>
<v>California</v>
<v>Richard LaValley</v>
<v>4000</v>
<v></v>
<v>10691.000000000000000</v>
<v>10694.000000000000000</v>
<v>0</v>
<v>2</v>
</o>
<o>
<v>SUGI 15</v>
<v></v>
<v>Opryland</v>
<v>Nashville</v>
<v>Tennessee</v>
<v>Tony L. Ona</v>
<v>3500</v>
<v></v>
<v>11048.000000000000000</v>
<v>11051.000000000000000</v>
<v>0</v>
<v>2</v>
</o>
<o>
<v>SUGI 16</v>
<v></v>
<v>Hilton Riverside & Towers</v>
<v>New Orleans</v>
<v>Louisiana</v>
<v>Martin J. Rosenberg</v>
<v>3500</v>
<v></v>
<v>11370.000000000000000</v>
<v>11373.000000000000000</v>
<v>0</v>
<v>1</v>
</o>
<o>
<v>SUGI 17</v>
<v></v>
<v>Hilton Hawaiian Village</v>
<v>Honolulu</v>
<v>Hawaii</v>
<v>E. Jeffrey Hutchinson</v>
<v>2500</v>
<v></v>
<v>11790.000000000000000</v>
<v>11793.000000000000000</v>
<v>1</v>
<v>2</v>
</o>
<o>
<v>SUGI 18</v>
<v></v>
<v>Hilton & Towers/Sheraton</v>
<v>New York</v>
<v>New York</v>
<v>Jerry L. Oglesby</v>

<v>3500</v>
<v></v>
<v>12182.000000000000000</v>
<v>12185.000000000000000</v>
<v>0</v>
<v>2</v>
</o>
<o>
<v>SUGI 19</v>
<v></v>
<v>Loews Anatole</v>
<v>Dallas</v>
<v>Texas</v>
<v>Peter Kretzman</v>
<v>2500</v>
<v></v>
<v>12518.000000000000000</v>
<v>12521.000000000000000</v>
<v>0</v>
<v>2</v>
</o>
<o>
<v>SUGI 20</v>
<v></v>
<v>Walt Disney World Dolphin</v>
<v>Orlando</v>
<v>Florida</v>
<v>Neil Howard</v>
<v>2600</v>
<v></v>
<v>12875.000000000000000</v>
<v>12878.000000000000000</v>
<v>1</v>
<v>2</v>
</o>
<o>
<v>SUGI 21</v>
<v></v>
<v>Chicago Hilton & Towers</v>
<v>Chicag0</v>
<v>Illinois</v>
<v>Paul Grant</v>
<v>3000</v>
<v></v>
<v>13218.000000000000000</v>
<v>13221.000000000000000</v>
<v>2</v>
<v>1</v>
</o>
<o>
<v>SUGI 22</v>
<v></v>
<v>San Diego Marriott Hotel & Marina</v>
<v>San Diego</v>
<v>California</v>
<v>Janet Bosomworth</v>
<v>3000</v>
<v>http://www2.sas.com/proceedings/sugi22/PROCEED.PDF</v>
<v>13589.000000000000000</v>
<v>13592.000000000000000</v>
<v>2</v>
<v>1</v>
</o>
<o>
<v>SUGI 23</v>
<v></v>
<v>Opryland Hotel</v>
<v>Nashville</v>
<v>Tennessee</v>
<v>Sally Goosetry</v>
<v>3000</v>
<v>http://www2.sas.com/proceedings/sugi23/Proceed.pdf</v>
<v>13960.000000000000000</v>
<v>13963.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 24</v>
<v></v>
<v>Fontainebleau Hilton Resort & Towers</v>
<v>Miami Beach</v>
<v>Florida</v>
<v>Mic Lajiness</v>
<v>3775</v>
<v>http://www2.sas.com/proceedings/sugi24/Proceed.pdf</v>
<v>14345.000000000000000</v>
<v>14348.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 25</v>
<v></v>
<v>Indiana Convention Center</v>
<v>Indianapolis</v>
<v>Indiana</v>
<v>Nancy Wilson</v>
<v>3000</v>
<v>http://www2.sas.com/proceedings/sugi25/PROCCED.pdf</v>
<v>14709.000000000000000</v>
<v>14712.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 26</v>
<v></v>
<v>Long Beach Convention Center</v>
<v>Long Beach</v>
<v>California</v>
<v>Frank Fry</v>
<v>3720</v>
<v>http://www2.sas.com/proceedings/sugi26/proceed.pdf</v>
<v>15087.000000000000000</v>
<v>15090.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 27</v>
<v></v>
<v>Walt Disney World Swan and Dolphin Resort</v>
<v>Orlando</v>
<v>Florida</v>
<v>Cyndie Gareleck</v>
<v>3676</v>
<v>http://www2.sas.com/proceedings/sugi27/Proceed27.pdf</v>
<v>15444.000000000000000</v>
<v>15447.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 28</v>
<v></v>
<v>Washington State Trade and Convention Center</v>
<v>Seattle</v>
<v>Washington</v>
<v>Art Carpenter</v>

<v>3036</v>
<v>http://www2.sas.com/proceedings/sugi28/Proceed.pdf</v>
<v>15794.000000000000000</v>
<v>15797.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 29</v>
<v></v>
<v>Le Palais des congres de Montreal</v>
<v>Montreal</v>
<v>Quebec</v>
<v>Duke Owen</v>
<v>2987</v>
<v>http://www2.sas.com/proceedings/sugi29/toc.html</v>
<v>16200.000000000000000</v>
<v>16203.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 30</v>
<v></v>
<v>Pennsylvania Convention Center</v>
<v>Philadelphia</v>
<v>Pennsylvania</v>
<v>Greg S. Nelson</v>
<v>3258</v>
<v>http://www2.sas.com/proceedings/sugi30/toc.html</v>
<v>16536.000000000000000</v>
<v>16539.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SUGI 31</v>
<v></v>
<v>Moscone West Convention Center</v>
<v>San Francisco</v>
<v>California</v>
<v>Kim LeBouton</v>
<v>4242</v>
<v>http://www2.sas.com/proceedings/sugi31/toc.html</v>
<v>16886.000000000000000</v>
<v>16889.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SAS Global Forum 2007</v>
<v></v>
<v>Walt Disney World Swan and Dolphin Resort</v>
<v>Orlando</v>
<v>Florida</v>
<v>Robert C. Fecht</v>
<v>3655</v>
<v>http://www2.sas.com/proceedings/forum2007/TOC.html</v>
<v>17272.000000000000000</v>
<v>17275.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
<o>
<v>SAS Global Forum 2008</v>
<v>http://support.sas.com/events/sasglobalforum/2008/index.html</v>
<v>Henry B. Gonzalez Convention Center</v>
<v>San Antonio</v>
<v>Texas</v>
<v>Warren E. Stinson</v>
<v>3794</v>
<v>http://www2.sas.com/proceedings/forum2008/TOC.html</v>
<v>17607.000000000000000</v>
<v>17610.000000000000000</v>
<v>2</v>
<v>1</v>
</o>
<o>
<v>SAS Global Forum 2009</v>
<v>http://support.sas.com/events/sasglobalforum/2009/index.html</v>
<v>Gaylord National Resort and Convention Center</v>
<v>Washington</v>
<v>DC</v>
<v>Lori Griffin</v>
<v>3328</v>
<v>http://support.sas.com/resources/papers/proceedings09/TOC.html</v>
<v>17978.000000000000000</v>
<v>17981.000000000000000</v>
<v>2</v>
<v>2</v>
</o>
</data>
<value_labels>
<vallab name='OnJansen'>
<label value='0'>None</label>
<label value='1'>Partial</label>
<label value='2'>Complete</label>
</vallab>
<vallab name='Season'>
<label value='1'>Winter</label>
<label value='2'>Spring</label>
<label value='3'>Summer</label>
<label value='4'>Fall</label>
</vallab>
</value_labels>
</dta>

APPENDIX 5 - THE XML MAP FILE (SGF_030_2010_XMLMAPUPDATED.MAP)

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ############################################################ -->
<!-- 2010-01-28T09:47:09 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 902000.3.6.20090116170000_v920 -->
<!-- ############################################################ -->
<!-- ###  Validation report                              ### -->
<!-- ############################################################ -->
<!-- XMLMap validation completed successfully. -->
<!-- ############################################################ -->
<SXLEMAP name="AUTO_GEN" version="1.2">

   <!-- ############################################################ -->
   <TABLE name="dta">
      <TABLE-DESCRIPTION>dta</TABLE-DESCRIPTION>
      <TABLE-PATH syntax="XPath">/dta</TABLE-PATH>

      <COLUMN name="dta_ORDINAL" ordinal="YES">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

   </TABLE>

   <!-- ############################################################ -->
   <TABLE name="header">
      <TABLE-DESCRIPTION>header</TABLE-DESCRIPTION>
      <TABLE-PATH syntax="XPath">/dta/header</TABLE-PATH>

      <COLUMN name="dta_ORDINAL" ordinal="YES">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="header_ORDINAL" ordinal="YES">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/header</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="ds_format">
         <PATH syntax="XPath">/dta/header/ds_format</PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="byteorder">
         <PATH syntax="XPath">/dta/header/byteorder</PATH>
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
         <LENGTH>4</LENGTH>
      </COLUMN>

      <COLUMN name="filetype">
         <PATH syntax="XPath">/dta/header/filetype</PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="nvar">
         <PATH syntax="XPath">/dta/header/nvar</PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>
```

```
    <COLUMN name="nobs">
      <PATH syntax="XPath">/dta/header/nobs</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="data_label">
      <PATH syntax="XPath">/dta/header/data_label</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>80</LENGTH>
    </COLUMN>

    <COLUMN name="time_stamp">
      <PATH syntax="XPath">/dta/header/time_stamp</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>16</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="descriptors">
    <TABLE-DESCRIPTION>descriptors</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors</TABLE-PATH>

    <COLUMN name="dta_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="typelist">
    <TABLE-DESCRIPTION>typelist</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/typelist</TABLE-PATH>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="typelist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/typelist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="type">
    <TABLE-DESCRIPTION>type</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/typelist/type</TABLE-PATH>

    <COLUMN name="typelist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/typelist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>
```

```
    <COLUMN name="type_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/typelist/type</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/descriptors/typelist/type/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="type">
      <PATH syntax="XPath">/dta/descriptors/typelist/type</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>5</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="varlist">
    <TABLE-DESCRIPTION>varlist</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/varlist</TABLE-PATH>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/varlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="variable">
    <TABLE-DESCRIPTION>variable</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/varlist/variable</TABLE-PATH>

    <COLUMN name="varlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/varlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="variable_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/varlist/variable</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/descriptors/varlist/variable/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="srtlist">
```

```
    <TABLE-DESCRIPTION>srtlist</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/srtlist</TABLE-PATH>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="srtlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/srtlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="sort">
    <TABLE-DESCRIPTION>sort</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/srtlist/sort</TABLE-PATH>

    <COLUMN name="srtlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/srtlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="sort_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/srtlist/sort</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/descriptors/srtlist/sort/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="fmtlist">
    <TABLE-DESCRIPTION>fmtlist</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/fmtlist</TABLE-PATH>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="fmtlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/fmtlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="fmt">
    <TABLE-DESCRIPTION>fmt</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/fmtlist/fmt</TABLE-PATH>

    <COLUMN name="fmtlist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/fmtlist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
```

```
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="fmt_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/fmtlist/fmt</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/descriptors/fmtlist/fmt/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="fmt">
      <PATH syntax="XPath">/dta/descriptors/fmtlist/fmt</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>10</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="lbllist">
    <TABLE-DESCRIPTION>lbllist</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/lbllist</TABLE-PATH>

    <COLUMN name="descriptors_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="lbllist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/lbllist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="lblname">
    <TABLE-DESCRIPTION>lblname</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/descriptors/lbllist/lblname</TABLE-PATH>

    <COLUMN name="lbllist_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/lbllist</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="lblname_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/descriptors/lbllist/lblname</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/descriptors/lbllist/lblname/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="lblname">
      <PATH syntax="XPath">/dta/descriptors/lbllist/lblname</PATH>
```

```
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="variable_labels">
    <TABLE-DESCRIPTION>variable_labels</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/variable_labels</TABLE-PATH>

    <COLUMN name="dta_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="variable_labels_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/variable_labels</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="vlabel">
    <TABLE-DESCRIPTION>vlabel</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/variable_labels/vlabel</TABLE-PATH>

    <COLUMN name="variable_labels_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/variable_labels</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="vlabel_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/variable_labels/vlabel</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="varname">
      <PATH syntax="XPath">/dta/variable_labels/vlabel/@varname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="vlabel">
      <PATH syntax="XPath">/dta/variable_labels/vlabel</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>2000</LENGTH>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="expansion">
    <TABLE-DESCRIPTION>expansion</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/expansion</TABLE-PATH>

    <COLUMN name="dta_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>
```

```
    <COLUMN name="expansion_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/expansion</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="char">
    <TABLE-DESCRIPTION>char</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/expansion/char</TABLE-PATH>

    <COLUMN name="expansion_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/expansion</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="char_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/expansion/char</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="name">
      <PATH syntax="XPath">/dta/expansion/char/@name</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="vname">
      <PATH syntax="XPath">/dta/expansion/char/@vname</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

    <COLUMN name="char">
      <PATH syntax="XPath">/dta/expansion/char</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>2000</LENGTH>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="data">
    <TABLE-DESCRIPTION>data</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/data</TABLE-PATH>

    <COLUMN name="dta_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="data_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/data</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE name="o">
    <TABLE-DESCRIPTION>o</TABLE-DESCRIPTION>
```

```
    <TABLE-PATH syntax="XPath">/dta/data/o</TABLE-PATH>

    <COLUMN name="data_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/data</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="o_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/data/o</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="v">
    <TABLE-DESCRIPTION>v</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/data/o/v</TABLE-PATH>

    <COLUMN name="o_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/data/o</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="v_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/data/o/v</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="v">
      <PATH syntax="XPath">/dta/data/o/v</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>2000</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="value_labels">
    <TABLE-DESCRIPTION>value_labels</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/value_labels</TABLE-PATH>

    <COLUMN name="dta_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="value_labels_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/value_labels</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="vallab">
    <TABLE-DESCRIPTION>vallab</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/value_labels/vallab</TABLE-PATH>

    <COLUMN name="value_labels_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/value_labels</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
```

```
    </COLUMN>

    <COLUMN name="vallab_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/value_labels/vallab</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="name">
      <PATH syntax="XPath">/dta/value_labels/vallab/@name</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>32</LENGTH>
    </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE name="label">
    <TABLE-DESCRIPTION>label</TABLE-DESCRIPTION>
    <TABLE-PATH syntax="XPath">/dta/value_labels/vallab/label</TABLE-PATH>

    <COLUMN name="vallab_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/value_labels/vallab</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="label_ORDINAL" ordinal="YES">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/dta/value_labels/vallab/label</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="value">
      <PATH syntax="XPath">/dta/value_labels/vallab/label/@value</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
    </COLUMN>

    <COLUMN name="label">
      <PATH syntax="XPath">/dta/value_labels/vallab/label</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>2000</LENGTH>
    </COLUMN>

  </TABLE>

</SXLEMAP>
```