PAPER 137-2008

Exporting SAS Datasets to DDI 3 XML files

Data, Metadata, and More Metadata

Larry Hoyle, Institute for Policy and Social Research, University of Kansas
Joachim Wackerow, *GESIS-ZUMA (Centre for Survey Research and Methodology, German Social Science Infrastructure Services)*

ABSTRACT

The Data Documentation Initiative (DDI) is an international effort to establish an XML-based standard for representing lifecycle information about social science research and similar data. DDI facilitates the automation of documentation and production systems for the delivery of social science data as well as for preservation and exchange. DDI 3 files also have the capacity to contain both metadata, and the data being described.

For some SAS users there will be a need to document existing SAS datasets via system-independent DDI files. This paper describes approaches to tools that can extract metadata and or data from an existing SAS dataset and add other metadata such as title, creator, funding agency etc. to create a DDI file. Two working implementations are compared, one uses PROC SQL and DATA steps, and the other uses a user-defined ODS tagset.

Other issues are discussed - such as the user interface for including that additional metadata (the who, what, when, where, why, and how); the potential need to document relationships like time series among tables; and the potential need to document referential constraints (foreign keys).

While some of the techniques involve advanced features, SAS users of all skill levels may find this paper of interest.

METADATA

Adequate metadata, information about any given set of data, is essential for the proper use and exchange of those data. Good metadata enables a researcher to get a full understanding of the data without any consultation between the data collector and the data user. In many situations where there is shared knowledge, it is not necessary to bundle a complete collection of metadata with a dataset when moving the data from one location to another or when using the data at a later time. This shared knowledge could be “common knowledge” or it could be metadata stored in a central or distributed metadata repository.

In other circumstances, though, it is necessary to have adequate metadata tightly bundled with data. One example might be data stored in an archive available for use by people with no promise of communication with the original producer of the data.

Linking of the data and metadata could be through storing them together in the same physical file. Alternatively, there could be references in the metadata to persistent Uniform Resource Identifiers (URIs), i.e. Uniform Resource Names. In the latter instance stability is an essential. Common web addresses (URLs) are not always immutable – domain names can change and paths within domains can and do change. Without tight bundling of data and metadata a data file can be redistributed without its associated metadata file possibly rendering it useless.

Metadata can be stored in many ways – codebooks on paper, codebooks in a somewhat structured digital format, or represented in some highly structured form such as in an XML file with a standardized schema. Having the metadata in a standardized format can make it machine-actionable, greatly increasing its utility.

DDI

DESCRIPTION

The Data Documentation Initiative (DDI) provides a standard for the compilation, presentation, and exchange of documentation for datasets in the social and behavioral sciences. The most recent version 3 of the DDI supports a rich and structured set of metadata elements that not only fully informs a potential data analyst about a given dataset but also facilitates computer processing of the data. DDI 3 supports the goal of capturing and preserving metadata on the full and continuing life-cycle of data from conception for preservation, distribution, application, and reuse in an open, non-proprietary system.

The DDI standard provides a rich and structured set of metadata elements and attributes that not only fully informs a potential data analyst about a given dataset but also facilitates computer processing of the data. Moreover, data producers will find that by adopting the DDI standard they can produce better and more complete documentation as a natural step in designing and fielding computer-assisted interviewing.

The new paradigm of DDI 3.0 embraces the full life cycle of the data from conception, through development of the data collection instrument, collection and cleaning of data, production of data products, distribution, preservation, and reuse or analysis of the data.

The structure of the new DDI 3.0 is designed to facilitate sharing concept schemes, question schemes, category and coding schemes, and variables schemes within organizations or throughout the social science research community. Comparison through direct inheritance as in the case of comparison-by-design or through the mapping of items like variables or categories allow capture of the harmonization processes used in creating integrated files in an uniform and machine-actionable way.

With a goal of capturing and preserving metadata on the full and continuing life-cycle of data from conception to preservation, distribution, application, and reuse, DDI 3.0 is providing the structural support needed to facilitate comparative survey work in a way that was previously unavailable in an open, non-proprietary system.

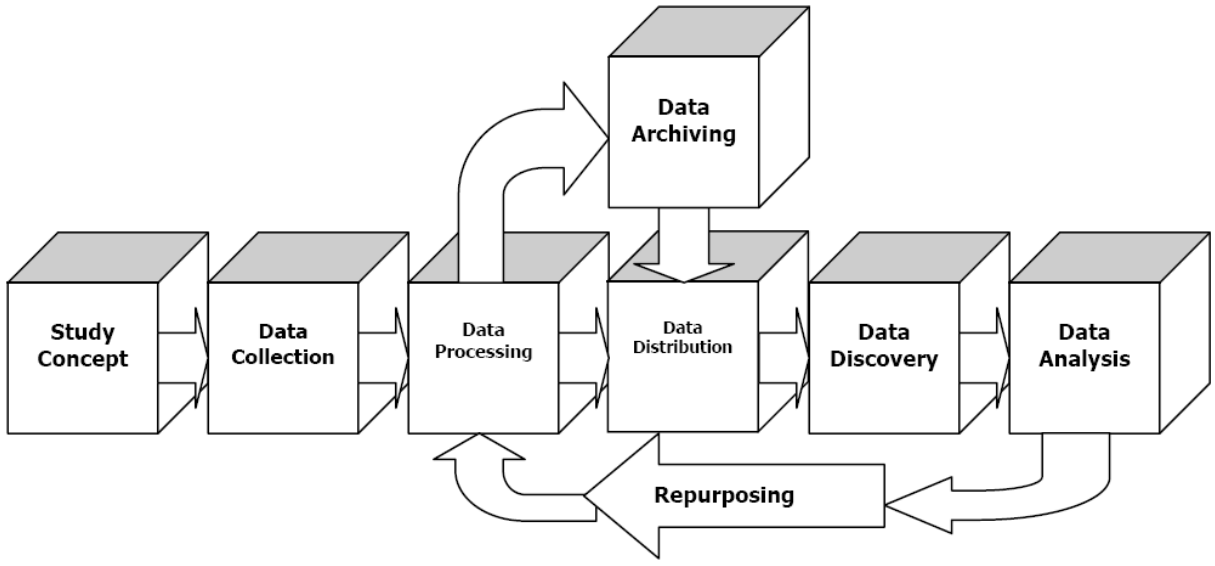
FEATURES OF DDI

- Metadata capture from planning and production to dissemination and analysis
- An underlying data model that permits the expression of the model in alternative technologies
- Coverage of more of the data life cycle, with an emphasis on data collection
- Modular design
- Enhanced support for multiple languages
- Support for variable comparison and harmonization
- Structured mechanisms for identification and versioning that enable the creation of registries like question banks
- Core HTML for formatting of unstructured text
- Elimination of redundancies through a new grouping model and an extensive set of reusable elements
- Grouping of study series for longitudinal and comparative research
- Capturing comparative information for the creation of harmonized data
- ISO/IEC 11179 compliant data registries such as question, variable, and concept banks
- Capability to create "DDI profiles" for specific uses
- Mechanism to carry data inline
- Alignment with other metadata standards, including Dublin Core (cross-domain information resource description), SDMX (time-series data), ISO/IEC 11179 (metadata registry), and FGDC and ISO 19115 (geographic standards)
- Extensibility

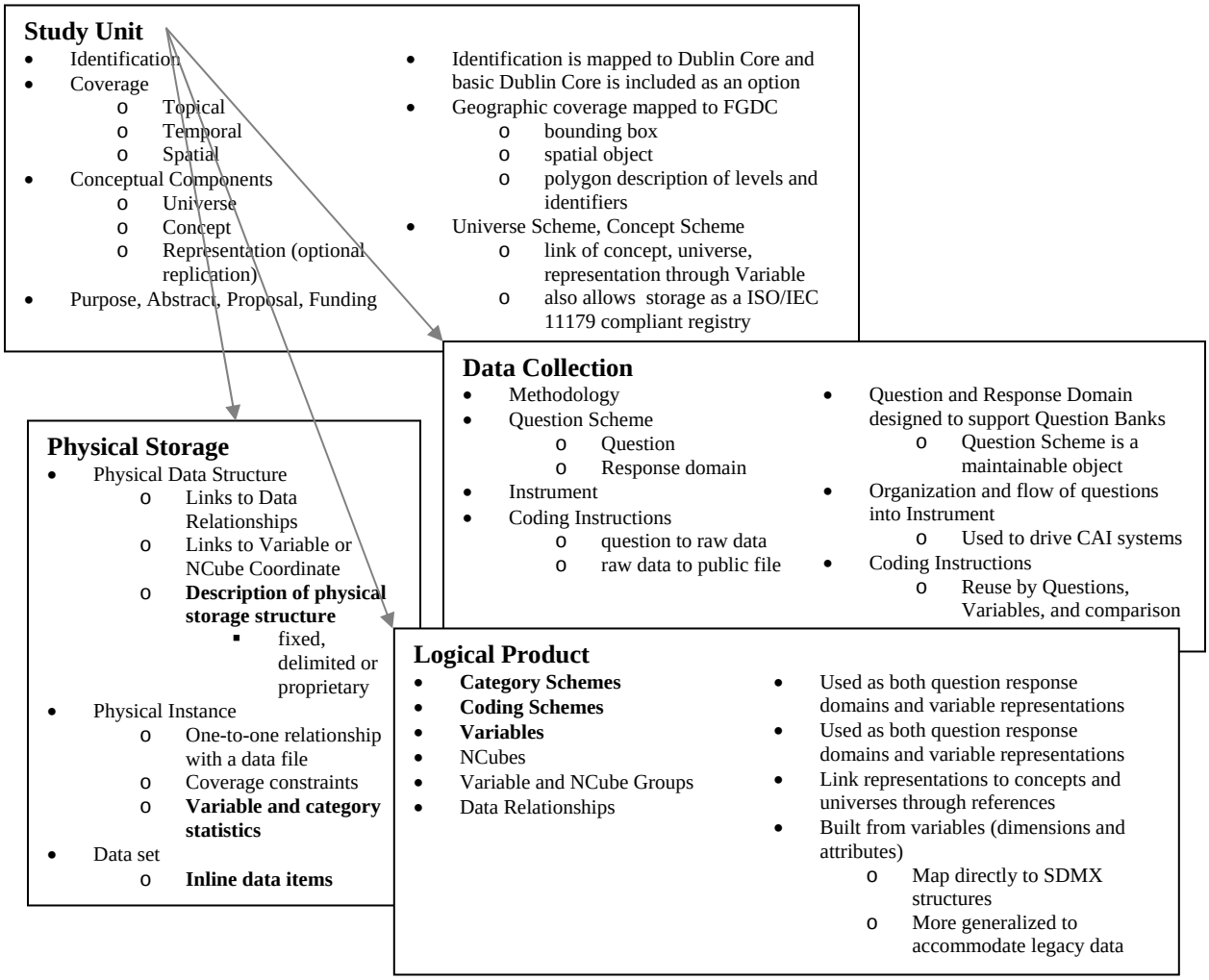
DDI 3 is described in a conceptual model which is also expressed in the Universal Modeling Language (UML). Modular XML Schemas are derived from the conceptual model. Many elements support computer processing – that is, it will go beyond being “human readable”, and move toward the goal of being “machine-actionable”.

DDI 3.0 is currently available as candidate draft in review process. The final release is announced for the second quarter of 2008. The tool fragments shown in this paper generate DDI files according the candidate draft version 1. The tools will be adapted to the final version of DDI 3.0.

LIFE-CYCLE OF DATA



CENTRAL DDI MODULES FOR A SINGLE STUDY WITH SECTIONS AND SHORT EXPLANATIONS
(metadata in SAS available in **bold**)



SAS METADATA MAPPED INTO THE DDI STRUCTURE

```
<ddi:DDIInstance xsi:schemaLocation="ddi:instance ...
 +<r:MaintainableID>
  <s:StudyUnit>
   +<r:MaintainableID>
   +<r:Citation>
   +<s:Abstract>
   +<r:UniverseReference>
   +<s:Purpose>
   +<cn:ConceptualComponent>
   +<d:DataCollection>
    <l:LogicalProduct>
     +<r:MaintainableID>
      <l:CategoryScheme>
        <r:MaintainableID>
        <r:ID>VA_CategoryScheme</r:ID>
        </r:MaintainableID>
        <l:Category>
          <r:VersionableID>
          <r:ID>VA_1</r:ID>
          </r:VersionableID>
          <r:Label>AU-Australia</r:Label>
```

user-defined format VA

label for value 1 of user-defined format VA

The DDI Alliance was formed with the intent of serving as standards development team for the DDI. The DDI Alliance is a self-sustaining membership organization whose members have a voice in the development of the DDI specification. Further information like the conceptual model and the XML field-level documentation can be found at the DDI Alliance web site.

METADATA IN THE SAS DATASET

A SAS dataset contains data of course, but it also contains metadata, information about the data. The dataset may contain some general information about the dataset in the dataset label. it may also contain information about each variable – a label, a format, an informat, a maximum length for a character variable, whether the dataset is sorted on the variable, whether the dataset is indexed on the variable, the scale and precision of the variable, whether the variable can be null. Some variable formats also imply a unit of measurement for the variable. The EURO format, for example, indicates that the variable represents currency in euros. The PERCENT format indicates that the data are actually stored as a proportion in spite of a label that might say something like "Percent Change". Integrity constraints may provide additional metadata about variables: e.g. whether they must have unique values, or not be null, or have some logical condition which must hold, or only have values contained in another table.

DICTIONARY TABLES

A convenient way to access these metadata in PROC SQL is through the DICTIONARY tables. The table at the right, generated by a `describe table` query, shows what information is available about each variable in DICTIONARY.COLUMNS. In a DATA step, functions like vformat are also available for extracting metadata about columns. The table below shows DICTIONARY.TABLES information that might be used outside of the SAS system.

```
libname char(8) label='Library Name',
memname char(32) label='Member Name',
memtype char(8) label='Member Type',
name char(32) label='Column Name',
type char(4) label='Column Type',
length num label='Column Length',
npos num label='Column Position',
varnum num label='Column Number in Table',
label char(256) label='Column Label',
format char(49) label='Column Format',
informat char(49) label='Column Informat',
idxusage char(9) label='Column Index Type',
sortedby num label='Order in Key Sequence',
xtype char(12) label='Extended Type',
notnull char(3) label='Not NULL?',
precision num label='Precision',
scale num label='Scale',
transcode char(3) label='Transcoded?'
```

PROC CONTENTS

Some information, like the name of a table referenced by a foreign key, is obtainable through PROC CONTENTS. The ODS OUTPUT facility makes it simple to capture this metadata.

```
Of interest outside SAS System
  memname char(32) label='Member Name',
  memlabel char(256) label='Dataset Label',
  memtype char(8) label='Member Type',       /* DATA or VIEW */
  crdate num format=DATETIME informat=DATETIME label='Date Created',
  modate num format=DATETIME informat=DATETIME label='Date Modified',
  nobs num label='Number of Physical Observations',
  typemem char(8) label='Dataset Type',  /* such as DATA, CORR, COV, SSPC, EST, or FACTOR */
  nvar num label='Number of Variables',
  nlobs num label='Number of Logical Observations',
  maxvar num label='Longest variable name',
  maxlabel num label='Longest label',

Possibly of interest outside SAS System
  libname char(8) label='Library Name',
  sorttype char(4) label='Sorting Type',
  sortchar char(8) label='Charset Sorted By',
```

DESIRABLE METADATA NOT IN THE DATASET

Just as a good journalist will cover the "who what when where why and how" of a story, adequate documentation for a dataset will include information that is not contained in the dataset itself or its associated formats. A dataset label is just not long enough to contain all of the information necessary to make data truly usable. This information might include a version date, title, creator, publisher, contributor, publication date, abstract, information on when where and how the data were collected, and the universe to which the data apply. The concepts underlying the columns might be described. Documenting relationships among the columns might also be necessary as when one column represents weights for other columns. In the case of survey data, the questions asked might be listed along with any instructions to the interviewers or respondents including skip and fill rules. Any sampling scheme used might be described. In many cases it would be impossible to draw any inferences from the data without this associated metadata. Other metadata might be useful in assisting search engines and mere mortals in finding the data. Examples include search keywords, and copyright information. Summary statistics might reassure those analyzing the data that they had correctly used any weighting or scale information about the data.

DDI CORE

The DDI standard allows for the structuring of all of these categories of metadata in a quite comprehensive manner. The DDI Alliance has also identified a minimal set of documentation, the "DDI Core", which should be included in a DDI file. The DDI Core was targeted as the set of metadata to be output from the sample applications described here.

CONVERTING METADATA FROM SAS TO DDI

DATA STEPS AND PROC SQL WITHIN A MACRO

Since an XML file is just text, one approach to creating a DDI file from a SAS dataset is to write the XML from a sequence of DATA steps. The metadata that are embedded in the dataset can be extracted either with functions like VFORMAT or with PROC SQL via the DICTIONARY tables. These metadata can be joined with a CNTLOUT dataset created by a PROC FORMAT step to capture user formatted values for those variables with permanently assigned user format values.

Metadata not in the dataset, like the creator of the dataset, can be captured as the values of macro variables. These macro variable values could be entered directly into code as arguments to macro invocations. They could also be entered via a Web form through a SAS/IntrNet application.

Writing the XML from DATA steps is relatively straightforward. Here is the code that writes the first few lines of a DDI file.

```
data _null_;
  length param $ &maxParamLength;
  file ddi Linesize=&DDILinesize;

  put "<?xml version=""1.0"" encoding=""UTF-8""?>";
  put "<ns1:DDIInstance xmlns:r=""ddi:reusable:3_0_CR""";
  put " xmlns:xhtml=""http://www.w3.org/1999/xhtml""";
  put " xmlns:dce=""ddi:dcelements:3_0_CR""";
```

XML SPECIAL CHARACTERS

There are five characters that cannot appear in the content of XML elements. These characters can, however, be escaped. So an ampersand '&' can be represented as the string '&' (without the single quotes). The safest approach then, is to scan all character values to be output to the DDI file with something like the function below.

```
                /* escape the five characters that cannot appear in PCDATA  */
%Macro Xescape(var=xxxx);
%do;
  prxchange('s/\"/&quot;/',1,
     prxchange('s/\''/&apos;/',1,
      prxchange('s/\>/&gt;/',1,
       prxchange('s/\</&lt;/',1,
        prxchange('s/\&/&amp;/',1,   &var.)))))
%end;
%Mend Xescape;

                /* put a macro variable value into the variable "param" */
                /* and then escape it */
%Macro GetParam(macroVar=ID);
%DO;
 param=symget("&macroVar");
 param=%Xescape(var=param);
%END;
%mend GetParam;
```

Here is an example of the Study Unit Creator element being written to the DDI file.

```
%GetParam(macroVar=SU_Creator)
  put "            <r:Creator>" param +(-1) "</r:Creator>";
```

UNITS OF MEASUREMENT

One way to capture unit of measurement information implied by SAS formats is to enter those relationships in a table. Here is a sample of the creation of such a table.

```
  /*  fmtname is the root name of the format (without the w.d) */
  /*  represents is the measurement unit that the format represents */
  /*  formatDescription the description of the format from SAS documentation */
data formatDocu;
length fmtname $ 32 represents $ 32 formatDescription $ 200;
input fmtname represents &:$char32. / formatDescription &:$char200.;
datalines;
DATE Date
Writes Date values in the form ddmmmyy or ddmmmyyyy
```

```
DATEAMP DateTime
Writes DateTime values in the form ddmmmyy:hh:mm:ss.ss with AM or PM
PERCENT Proportion
Writes numeric values as percentages
```

GATHERING THE METADATA

A sequence of SQL queries can gather the metadata contained in the target dataset.

```
proc sql;
 /*  extract the column metadata and escape characters where necessary */
  create table myColumnsA as
   select *,
          prxchange('s/\d*\.\d*//',1,format) as fmtName,
          %Xescape(var=name) as xname,
          %Xescape(var=label) as xlabel
   from dictionary.columns
   where libname=upcase("&lib") and memname=upcase("&dataset")
   order by name;
/* join in the information from the documentation for formats */
  create table myColumns as
   select MyColumnsA.*,
          represents,
          %Xescape(var=formatDescription) as formatDescription
   from MyColumnsA left join formatDocu
   on MycolumnsA.fmtname=formatDocu.fmtname
   order by name;
/* extract the format metadata for our columns */
  create table userFormats as
   select myFormats.*
    from myformats,myColumns
    where myColumns.fmtName=myFormats.fmtname and myFormats.type in ("N","C");
/*  make a unique list of the labels from the formats useful for concepts */
  create table uniqueLabels as
   select distinct fmtname, label
    from userformats
    order by fmtname, label;
/* join in user format names */
  create table MyColumns2a as
   select MyColumns.*, uFmtName
   from MyColumns left join (select distinct fmtname as uFmtName label="User Format
Name" from userformats) as uf
       on  MyColumns.fmtname=uf.ufmtname
   order by name;
/*Capture metadata from integrity constraints */
ods output Contents.DataSet.IntegrityConstraints=work.IC1;
proc contents data=&lib..&dataset;
run;
ods output close;
/* link the integrity constraints to the column name */
proc sql;
 create table ColumnConstraints as
 select dict.Column_name,
        ic1.constraint, ic1.Type, ic1.Variables, ic1.WhereClause,
        ic1.Reference, ic1.OnDelete, ic1.OnUpdate
 from ic1,DICTIONARY.CONSTRAINT_COLUMN_USAGE as dict
 where dict.table_catalog=upcase(symget("lib")) and
       dict.table_name=upcase(symget("dataset")) and
       ic1.constraint=dict.constraint_name
order by Column_name, type;
```

Once the constraint metadata are tabulated, they can be concatenated into a single note for each variable and joined with the rest of the metadata. Potentially, the where clauses of the “check” constraints could be parsed to yield lists or ranges of acceptable values. Foreign key references could also be followed to list acceptable values.

ENTERING METADATA

Here is an example of entering the metadata for the original question that was associated with the ID variable in the dataset. Note that this is not necessarily the same text as in the SAS variable label. The NRSTR macro function is used to allow the entry of characters that might otherwise be interpreted as being part of the SAS program.

```
%QuestionItem(QuestionItem_ID= %NRSTR(sex),
              QuestionItem_Text= %NRSTR(What is the sex of the respondent?),
              QuestionItem_CodeScheme= %NRSTR(Codes_SEX) ) ;
```

REPRESENTING THE DATA

The DDI schema allows for either embedding data in the DDI file itself or documenting the layout of the data in an external file. The markup for data within a DDI file is rather voluminous so the designers of DDI view the embedded data option as applying mostly to small datasets. Embedding data in a DDI file might not be the best choice for data exchange, particularly for closely related entities who can easily exchange separate metadata. For archiving data though, a major benefit of embedding data is that the data and metadata cannot become separated.

The DATA step approach presented in this paper embeds the data in the DDI <ds:DataSet> element. The application could be extended to write the data to a separate file and add documentation of the layout of that file to the DDI file.

Code for this DATA Step approach is available at: http://www.ipsr.ku.edu/ksdata/sashttp/SGF2008/.

A USER-DEFINED TAGSET FOR THE DDI FORMAT

OUTPUT DELIVERY SYSTEM (ODS) AND USER-DEFINED TAGSETS

ODS allows output from SAS procedures to be exported to a variety of file formats like PDF, HTML, RTF, etc. The creation of user-defined ODS schemes is possible by template definitions. This way markup language tagsets can be built for custom XML output formats like DDI by `PROC TEMPLATE`. A tagset is based on an event-driven model; it doesn't work like a program in a procedural language. A SAS procedure output can be understood as a sequence of events. An event is an action that can be used to generate custom output. A tagset is a collection of event definitions. The event definition can include `start` and `finish` sections for different actions for finer control. Not every event definition must correspond to a real event. This kind of helper event definitions can be seen as subroutines. An event can execute another event. This way the tagset can be organized in more easily understandable units. The writing of a tagset requires the learning of the related definition language.

The usage of user-defined tagsets is well integrated in ODS. The user opens a file for ODS DDI destination, executes the required procedures, and closes the destination. By selection of the procedures, their options and be defining options for the DDI tagset the user has a finer control over the DDI output.

PREPARING THE METADATA FOR THE DDI TAGSET

The user-defined formats associated with a SAS data set need some preparation. Similar to the step in the described macro above a temporary data set or table (`userFormats`) must be built for the user-defined formats. Only the user-defined formats should be exported which are actually used by the variables.

USING THE DDI TAGSET

The DDI tagset works with some SAS procedures to export the SAS metadata into the DDI format. These are basically the procedures `CONTENTS`, `REPORT`, `PRINT`, and `FREQ` to output information about a SAS data set.

```
/* specifiying the DDI tagset as ODS destination and opening a file */
ods tagsets.DDI file='ddi.xml' encoding='utf-8';
/* using selected SAS procedures to output information about a SAS data set */
proc contents data=library.mySASdata;
run;
proc report data=userFormats;
run;
proc freq data=library.mySASdata;
run;
/* closing the ODS destination for DDI */
ods tagsets.DDI close;
```

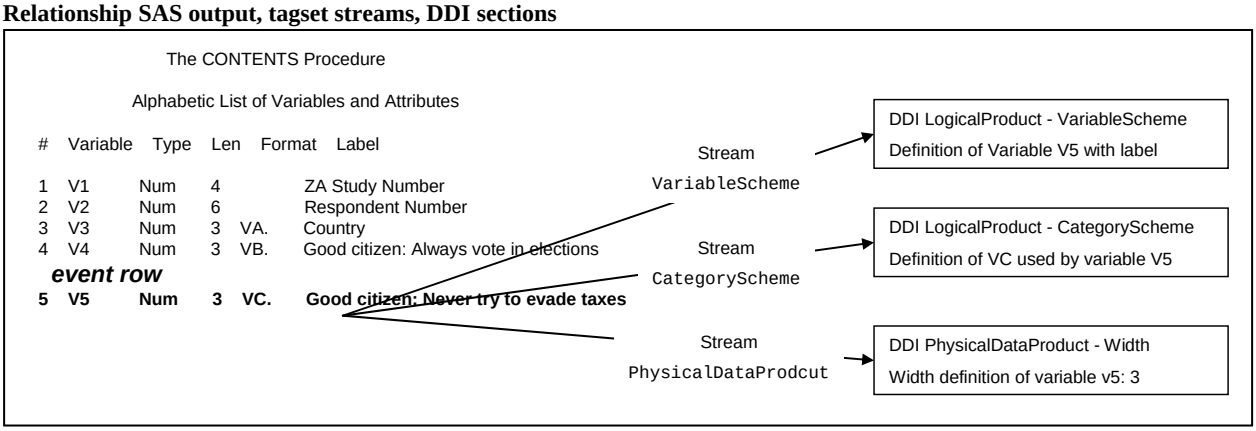
DETAILS OF THE DDI TAGSET

Information about variables, codes, data formats, and user-defined formats are stored in the DDI structure in various modules and schemes. For example it is required to use the variable names at several places in DDI: variable scheme, category scheme, concept scheme, etc. An ODS template or tagset runs only once through the SAS output of events. To get the same variable names into various locations specific streams can be used like one stream for each of the DDI schemes mentioned above. A stream can be understood as a memory buffer which can be opened and closed when needed. With an opened stream the regular output generated by `put` does not go to a file but to the specified stream. The content of a stream can then be written

asynchronously to the file when needed. This way the information of the SAS output can be reorganized in the way how the DDI XML format is structured.

The events “row” and “data” containing variable information write various streams and result in related DDI definitions.

Relationship SAS output, tagset streams, DDI sections




```
define event _Variable ;
  start:
     flush ;
     open LogicalProduct ;
     putl '<l:Variable>' ;
     flush ;
     close ;
  finish:
     flush ;
     open LogicalProduct ;
     ndent ;
     putl '<r:IdentifiableID>' ;
     ndent ;
     put '<r:ID>Variable_' ;
     put $variable_name ;
     putl '</r:ID>' ;
     put '<r:Name>' ;
     put $variable_name ;
     putl '</r:Name>' ;
     xdent ;
     putl '</r:IdentifiableID>' ;
     put '<r:Label>' ;
     put $variable_label ;
     putl '</r:Label>' ;
     putl '<l:ConceptReference>' ;
     ndent ;
     putl '<r:Reference>' ;
     ndent ;
     put '<r:ID>Concept_' ;
     put $variable_name ;
     putl '</r:ID>' ;
     xdent ;
     putl '</r:Reference>' ;
     xdent ;
     putl '</l:ConceptReference>' ;
     xdent ;
     putl '</l:Variable>' ;
     flush ;
     close ;
end;
```

The event definition (here helper type for variable) shown at left, generates the DDI structure (here fragment for one basic variable definition) shown below. This DDI structure is written to the LogicalProduct stream which is later written to the appropriate place in the DDI file.

```
<l:Variable>
  <r:IdentifiableID>
    <r:ID>Variable_V3</r:ID>
    <r:Name>V3</r:Name>
  </r:IdentifiableID>
  <r:Label>Country</r:Label>
  <l:ConceptReference>
    <r:Reference>
      <r:ID>Concept_V3</r:ID>
    </r:Reference>
  </l:ConceptReference>
</l:Variable>
```

The five special XML characters which are not allowed in XML content must be escaped as with the macro approach in a previous section. This can be achieved in a ODS tagset by the following mapping definition:

```
map = %nrstr('<>&"''') ;
mapsub =
%nrstr('/&lt;/&gt;/&amp;/&quot;/&apos;/') ;
```

Complete code for the DDI tagset approach to writing a DDI file is available at:
http://www.gesis.org/en/software/ddi/tools/sas_ods_ddi_tagset/

ODS WITH DEFAULT TAGSET PLUS XSLT TRANSFORMATION

This approach uses the SAS default tagset to generate an XML representation of the SAS metadata. The SAS-specific XML structure can be transformed by an XSLT processor to a DDI XML file according to an XSLT stylesheet. The advantage of this approach is to use a standard language (XSLT) for building the stylesheet. The two steps (exporting according the SAS default tagset and transforming to a DDI file by an XSLT processor) can be combined by using a SAS pipe as input for the XSLT processor (here XALANC).

```
filename xsltproc pipe 'xalanc -t -i 2 -o DDI.xml – SASDefaultTagset2DDI.xslt';
ods tagsets.Default file=xsltproc encoding='utf-8';
/* using selected SAS procedures to output information about a SAS data set */
proc contents data=library.mySASdata;
run;
...
/* closing the ODS destination for Default */
ods tagsets.Default close;
```

The messages of the XSLT processor are shown in the SAS log, like the consumed time. Additional software – the XSLT processor - is necessary for the application of the stylesheet. Instead of XALANC any other XSLT processor can be used.

No investment in a user-defined tagset is necessary in using the available SAS default tagset. Learning of the specific SAS tagset language does not apply. XSLT allows a powerful transformation in using a standard XML standard language.

In principle this approach produces some overhead. In using the pipe the output of the default tagset is buffered in the memory. The XSLT processor builds a DOM representation in memory before applying the stylesheet. This can be a limitation in processing SAS data sets with a huge number of variables or cases (when frequencies or the data itself are required). On the other hand this approach is flexible. It can be used as a two-step process or a virtual one-step process – like shown above, and once the stylesheet exists it can be adapted without large effort.

OTHER APPROACHES TO GATHER SAS METADATA: ODBC AND ADO/OLE DB

ODBC and ADO/OLEDB provide means to read SAS data sets on computers without the SAS system installed. This can be an advantage in special circumstances. SAS provides free software to build related tools as Windows-only solution.

Open Database Connectivity (ODBC) provides a standard software API to access data stores. Standard SQL queries can be used to export information about SAS data sets. For Java programmers a bridge to JDBC (Java Database Connectivity) exists. For solutions into this direction see Zhang, 2003.

Microsoft OLE DB (Object Linking and Embedding, Database) is another way to access data stores. It can be understood as an updated and generalized ODBC. Microsoft ActiveX Data Objects (ADO) provides an easy programmatic interface (also for Java) to use OLE DB. It is a layer on top of OLE DB. ADO refers only to a subset of OLE DB. For example it allows reading a SAS dataset but not submitting SAS code. The SAS local data provider supports access to SAS data sets. This doesn't require the SAS system installed, but doesn't allow SQL queries. For solutions into this direction see Churchill.

DISCUSSION / COMPARISON

Any approach to generating a file containing the metadata from a SAS dataset will involve first gathering the metadata either through some combination of the dictionary tables, function calls, and PROC CONTENTS. The ultimate choice of which method to use may depend on the context in which the process will be used. Whatever the approach, there will be a need to supplement the metadata present in the dataset with other metadata. Some of these metadata are available at the time in which a dataset is first designed such as the purpose of the dataset and the original designer. Other metadata become available during the collection of the data – dates and times of events, who worked on data collection. Some metadata may change even after the data are finalized as in the URI of copies of the dataset.

An approach like the one presented here using Data Steps and Proc SQL within a macro could be adapted to a SAS/Intrnet application which allowed a user to submit a dataset and fill in a form for the metadata not available in the dataset - the business metadata or the resource discovery metadata. Another potential approach could use stored processes and Enterprise Guide. The "SAS only" approaches would be awkward for handling metadata with text descriptions longer than the limit for a character variable value. In those situations modifying the DDI file outside of SAS may be more practical.

In a context where there are external facilities for combining the technical metadata (from SAS files) with other metadata, such as where there are other DDI tools already in place or where users are comfortable using an XML editor, a SAS-centric user interface may not be wanted.

The macro approach seems to be very flexible for expansion by the experienced SAS programmer. In contrast the tagset approach can be integrated in a ODS tagset library and can be used in a similar way as other output tagsets.

USER INTERFACE ISSUES

Some of elements in a DDI file could contain a very long character string. Entering such text into macro invocation arguments could be awkward. Implementing a Web based entry form, some other application such as a .NET application, or entering the data into an external table might work better for long content.

The DDI specification also allows for the entry of XHTML as content for some elements. This subset of XHTML is limited to structural layout elements. Revisions to the metadata once a DDI file is created might also best be handled by DDI specific tools outside of SAS.

BEYOND THE SINGLE DATASET

As mentioned before, the applications described here do not allow for the description of relationships among datasets. In many cases this information would have to come from outside the SAS system and it might be best to use specialized external tools to add that information.

In some cases important information about relationships among datasets does lie within the SAS system. A trivial example would be datasets that reside in the same library. A more important case would be where there are foreign key relationships defined. A future enhancement to these applications might track down all of these relationships and document the whole set of related tables together.

DDI files are also capable of documenting nCubes (multidimensional tables). Implementing that is left as an exercise for the reader.

COMPLETING THE ROUND TRIP

The SAS system has the capability of reading a DDI file via the XML libname engine. An XMLMAP file for DDI files can be used to map the hierarchical layout of the DDI file into a set of relational tables. Formats can be reconstructed from these tables. For embedded data this would complete the round trip from SAS to DDI to SAS again. In the case of a DDI file documenting an external data file, the information in the relational tables can be used to construct and execute a SAS program to read the external data into SAS (see Hoyle, 2005).

KEYWORDS:

DDI, metadata, interoperability, XML/XSLT, archive, ODS tagset

REFERENCES

Churchill, Alan *Step by Step Guide for Developing Your First .NET SAS Project*
http://saviandata.savian.net/papers/Step%20by%20Step%20Guide%20for%20Developing%20Your%20First.doc

DDI Alliance; Data Documentation Initiative http://www.ddialliance.org/

Gebhart, Eric *The Beginners Guide to ODS MARKUP: Don't Panic!*
SAS Users Group International Conference 31, (SUGI31), San Francisco, California, March 2006.
http://www2.sas.com/proceedings/sugi31/263-31.pdf

Hoyle, Larry *Using XML Mapper and XMLMAP to Read Data Documented by Data Documentation Initiative (DDI) Files* SAS
SAS Users Group International Conference 30, (SUGI30), Philadelphia, Pennsylvania, April 2005.
http://www2.sas.com/proceedings/sugi30/099-30.pdf

Lund, Pete *A Quick and Easy Data Dictionary Macro*
SAS Users Group International Conference 27 (SUGI27), Orlando, Florida, April 2002.
http://www2.sas.com/proceedings/sugi27/p099-27.pdf

Miller, Ken and Mary Vardigan *How Initiative Benefits the Research Community - the Data Documentation Initiative*
http://www.ddialliance.org/DDI/papers/miller.pdf

Thomas, Wendy; Gregory, Arofan; Gager, Jack; Kuo, I-Lin; Wackerow, Joachim; Nelson, Chris *Data Documentation Initiative (DDI), Technical Specification, Part I, Overview, For Candidate Review (Version 3.0)*
http://www.ddialliance.org/DDI/ddi3/DDI_3.0_Part_I_Overview_cr.pdf

Thomas, Wendy *Data Documentation Initiative*
10th Annual Open Forum for Metadata Registries, New York, July 2007
http://metadataopenforum.org/index.php?id=21,72,0,0,1,0

Zhang, Lei *Datamapper: A Documentation Generator for SAS Metadata*
Pharmaceutical Industry SAS Users Group Conference Proceedings (PharmaSUG 2003), Miami, Florida, May 2003.
http://www.lexjansen.com/pharmasug/2003/applicationsdevelopment/ad095.pdf

ACKNOWLEDGEMENTS

Many thanks to Wendy Thomas for helping with the DDI description.

CONTACT INFORMATION

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

LARRY HOYLE
Institute for Policy and Social Research
University of Kansas
1541 Lilac Road, 607 Blake
Lawrence, KS 66044-3177
USA
+1 785-864-9110
LarryHoyle@ku.edu
www.ipsr.ku.edu

JOACHIM WACKEROW
GESIS-ZUMA (Centre for Survey Research and Methodology, German Social Science Infrastructure Services)
B2, 1
68159 Mannheim
Germany
+49 621 1246 262
joachim.wackerow@gesis.org
www.gesis.org