DRAFT Using XMLMAP to Read Data Documented by a Data Documentation Initiative (DDI) File DRAFT

Larry Hoyle Policy Research Institute, The University of Kansas

Abstract

Data Documentation Initiative files are XML "codebooks" for social science data. SAS XMLMAP provides the capability to read these metadata into a collection of SAS datasets. SAS code can then be produced to read the data documented by the DDI file. This paper also describes how the XML Atlas tool was useful in developing the XMLMAP for this project.

What is DDI?

The front page for the Data Documentation Initiative (DDI), http://www.icpsr.umich.edu/DDI/, describes DDI as "an international effort to establish a standard for technical documentation describing social science data". DDI provides a means to structure metadata as an XML document.
The DDI XML format is defined by a Document Type Definition (DTD) available at http://www.icpsr.umich.edu/DDI/users/dtd/Version2-0.dtd.zip. The top level of this metadata file is defined as

```
<!ELEMENT codeBook     (docDscr*
                 , stdyDscr+
                 , fileDscr*
                 , dataDscr*
                 , otherMat*)                    >
```

The elements of the DDI file within the root element "codebook" describe:

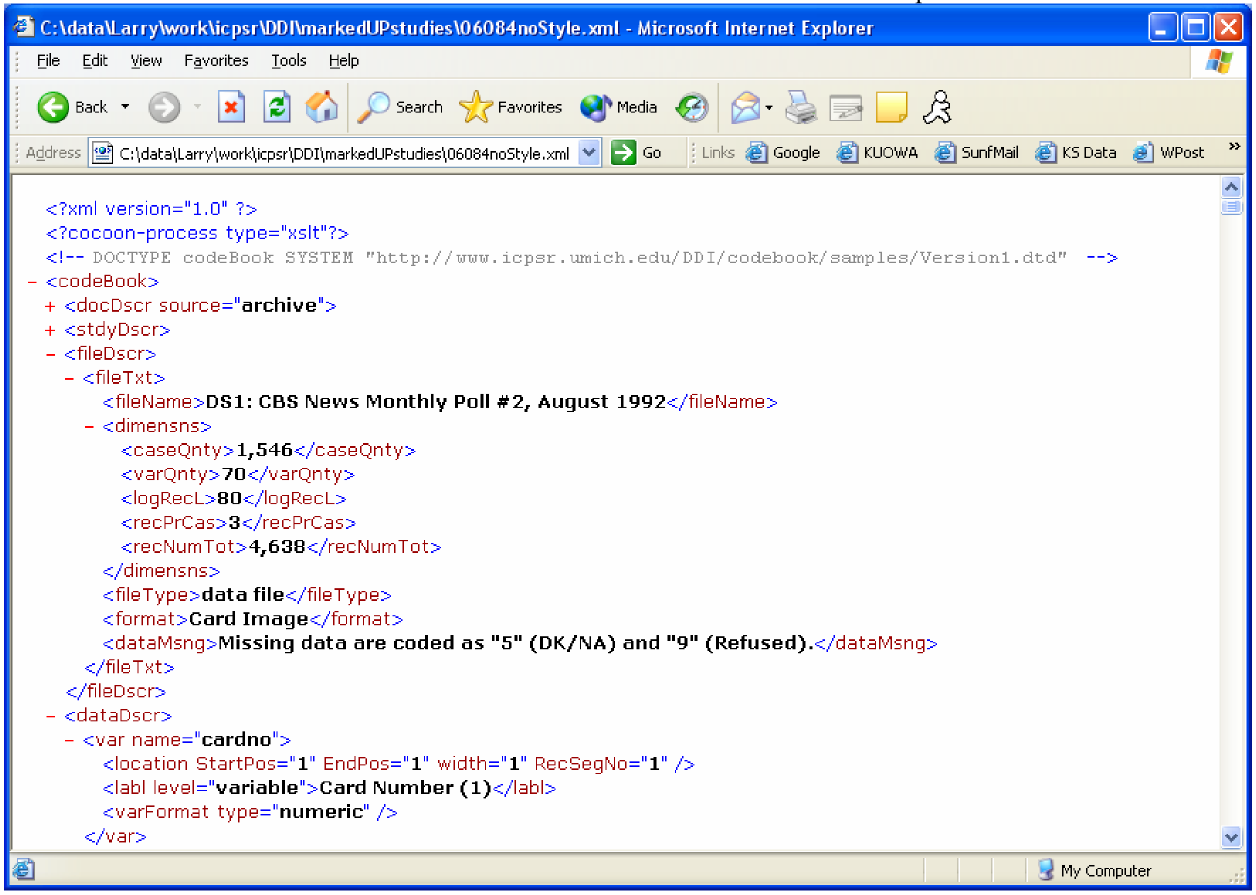
- *docDscr* - the DDI document itself
  - can include a citation, status, source, notes and more
- *stdyDsc* - the dataset (the study) described by the DDI file
  - can include a citation, study information, method, terms of use and more
- *fileDscr* - the data file
  - can include a URI for locating the file; information about content, structure, format, number of records, record length and more.
- *dataDscr* - the variables within the data file
  - can include information about variable groups, ncube groups as well as information about variables – name, weight, question, location(start position, end position, width, record segment) and more
- *otherMat* - other material
  - could include questionnaires, SAS code to read the data, maps etc.

A Sample DDI file

This paper will use the codebook for ICPSR study 6084 – "CBS News Monthly Poll #2, August 1992" as the example DDI file to be read. The DDI file is available at: http://www.icpsr.umich.edu/DDI/samples/index.html. The first few lines of the XML DDI file are:

```
<?xml version="1.0"?>
<?xml-stylesheet href="../XSL/codebook.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
<!--DOCTYPE codeBook SYSTEM "http://www.icpsr.umich.edu/DDI/codebook/samples/Version1.dtd"-->
<codeBook>
  <docDscr source="archive">
          <citation>
                  <titlStmt>
                    <titl>CBS News Monthly Poll #2, August 1992</titl>
                    <altTitl>August National Poll II, Republican National
                            Convention</altTitl>
                    <IDNo agency="ICPSR">6084</IDNo>
                  </titlStmt>
```

These lines reveal the hierarchical nature of the file. The "root" of the file is a <*codebook*> tag, under which lies the <*docDscr source="archive"*> tag, and within that the <*citation*> tag. More tags are, in turn, nested within the citation. Removing the <*?xml-stylesheet* tag in line two and opening up the file in MS Internet Explorer 6 displays the XML file as an outline. Here is a view of that file with some of the elements collapsed:


C:\data\Larry\work\icpsr\DDI\markedUPstudies\06084noStyle.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\data\Larry\work\icpsr\DDI\markedUPstudies\06084noStyle.xml Go Links Google KUOWA SunfMail KS Data WPost
My Computer

```
  <?xml version="1.0" ?>
  <?cocoon-process type="xslt"?>
  <!-- DOCTYPE codeBook SYSTEM "http://www.icpsr.umich.edu/DDI/codebook/samples/Version1.dtd"  -->
- <codeBook>
  + <docDscr source="archive">
  + <stdyDscr>
  - <fileDscr>
    - <fileTxt>
        <fileName>DS1: CBS News Monthly Poll #2, August 1992</fileName>
      - <dimensns>
          <caseQnty>1,546</caseQnty>
          <varQnty>70</varQnty>
          <logRecL>80</logRecL>
          <recPrCas>3</recPrCas>
          <recNumTot>4,638</recNumTot>
        </dimensns>
        <fileType>data file</fileType>
        <format>Card Image</format>
        <dataMsng>Missing data are coded as "5" (DK/NA) and "9" (Refused).</dataMsng>
      </fileTxt>
    </fileDscr>
  - <dataDscr>
    - <var name="cardno">
        <location StartPos="1" EndPos="1" width="1" RecSegNo="1" />
        <labl level="variable">Card Number (1)</labl>
        <varFormat type="numeric" />
      </var>
```

What is SAS XMLMap?

In order to retrieve data from a hierarchical XML file into SAS datasets there has to be a way to specify a "mapping" between the hierarchy and a rectangular table. The SAS XMLMap file is that mapping.
The SAS XMLMap file is an XML file that describes which elements of an XML target file define rows and which elements define columns. In the example below a SAS dataset named *fileInfo* is created with one row for each *fileTxt* element that is nested in a *fileDscr* element. The dataset has two columns, one taken from the *fileName* element and one from the *caseQnty* element.

```
<!-- 2003-05-18T12:31:09.041 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XMLAtlas, Version 9.0.1 -->

<SXLEMAP version="1.1" name="SXLEMAP">
<TABLE name="fileInfo">
   <TABLE-PATH>/codeBook/fileDscr/fileTxt</TABLE-PATH>

   <COLUMN name="fileName">
     <PATH>/codeBook/fileDscr/fileTxt/fileName</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>42</LENGTH>
   </COLUMN>

   <COLUMN name="caseQnty">
     <PATH>/codeBook/fileDscr/fileTxt/dimensns/caseQnty</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>5</LENGTH>
   </COLUMN>

  </TABLE>

</SXLEMAP>
```

The following SAS program prints the dataset described by the SAS XMLMap file above.

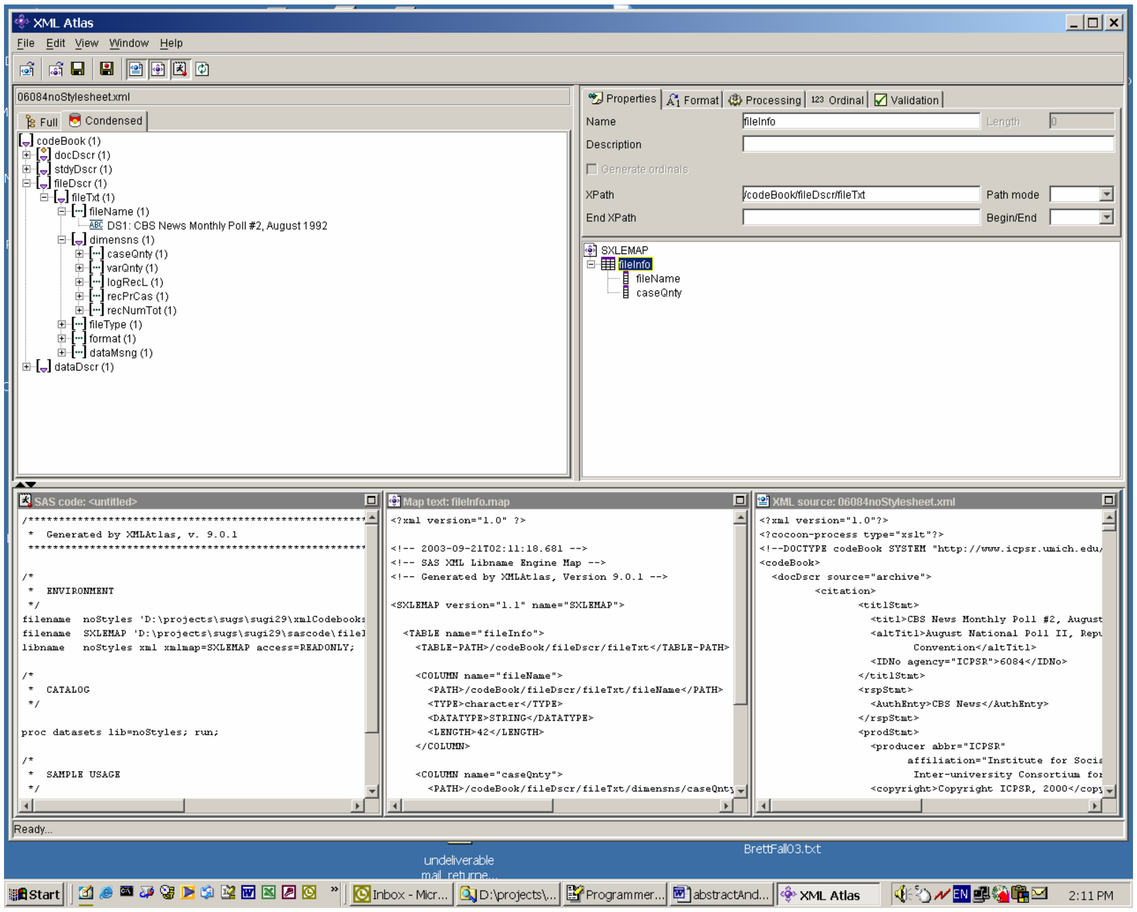
```
/* read a two column table from DDI file for ICPSR 6084 */
filename  noStyles 'D:\projects\sugs\sugi29\xmlCodebooks\06084noStylesheet.xml';
filename  SXLEMAP 'D:\projects\sugs\sugi29\sascode\fileInfo.map';
libname   noStyles xml xmlmap=SXLEMAP access=READONLY;

proc print data=nostyles.fileInfo;
run;
```

The resulting output is:

Obs	fileName	caseQnty
1	DS1: CBS News Monthly Poll #2, August 1992	1,546

Viewing the sample DDI file in XML Atlas

SAS XML Atlas in SAS 9 is a graphical user interface for creating SAS XMLMap files. It allows drag and drop creation of the XMLMap file and automatically creates the SAS file to reference it. Using Atlas was an easy way to create the XMLMap file to read the DDI file. Here is an Atlas view of the files we've been working with so far. XML elements can be dragged from the upper left pane to the upper right pane, creating the design of the tables to be extracted. The bottom center pane shows the XMLMap file that Atlas generates and the bottom left pane shows sample SAS code that will access that XMLMap file.



```
codeBook (1)
  docDscr (1)
  stdyDscr (1)
  fileDscr (1)
    fileTxt (1)
      fileName (1)
        DS1: CBS News Monthly Poll #2, August 1992
      dimensns (1)
        caseQnty (1)
        varQnty (1)
        logRecL (1)
        recPrCas (1)
        recNumTot (1)
      fileType (1)
      format (1)
      dataMsng (1)
  dataDscr (1)
```

Name: fileInfo
Description:
XPath: /codeBook/fileDscr/fileTxt
End XPath:

```
SXLEMAP
  fileInfo
    fileName
    caseQnty
```

SAS code: <untitled>

```
/*****************************************************************
 *  Generated by XMLAtlas, v. 9.0.1
 *****************************************************************

/*
 *  ENVIRONMENT
 */
filename  noStyles 'D:\projects\sugs\sugi29\xmlCodebooks
filename  SXLEMAP 'D:\projects\sugs\sugi29\sascode\fileI
libname   noStyles xml xmlmap=SXLEMAP access=READONLY;

/*
 *  CATALOG
 */

proc datasets lib=noStyles; run;

/*
 *  SAMPLE USAGE
 */
```

Map text: fileInfo.map

```
<?xml version="1.0" ?>

<!-- 2003-09-21T02:11:18.681 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XMLAtlas, Version 9.0.1 -->

<SXLEMAP version="1.1" name="SXLEMAP">

  <TABLE name="fileInfo">
    <TABLE-PATH>/codeBook/fileDscr/fileTxt</TABLE-PATH>

    <COLUMN name="fileName">
      <PATH>/codeBook/fileDscr/fileTxt/fileName</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>42</LENGTH>
    </COLUMN>

    <COLUMN name="caseQnty">
      <PATH>/codeBook/fileDscr/fileTxt/dimensns/caseQnty
```

XML source: 06084noStylesheet.xml

```
<?xml version="1.0"?>
<?cocoon-process type="xslt"?>
<!--DOCTYPE codeBook SYSTEM "http://www.icpsr.umich.edu/
<codeBook>
  <docDscr source="archive">
         <citation>
                <titlStmt>
                  <titl>CBS News Monthly Poll #2, August
                  <altTitl>August National Poll II, Repu
                         Convention</altTitl>
                  <IDNo agency="ICPSR">6084</IDNo>
                </titlStmt>
                <rspStmt>
                  <AuthEnty>CBS News</AuthEnty>
                </rspStmt>
                <prodStmt>
                  <producer abbr="ICPSR"
                        affiliation="Institute for Socia
                         Inter-university Consortium for
                  <copyright>Copyright ICPSR, 2000</copy
```

Extracting SAS datasets from the hierarchical DDI file

Only a small subset of the elements in the DDI file are actually needed to write a SAS program to read the data file, but it is useful to read additional elements to document the code. For this example, Eight SAS datasets were created, with the following elements in each.

SAS Dataset	Column Name (XML DDI element name)
ABSTRACT	abstract, source
DATADSCRVAR	name, StartPos, EndPos, width, RecSegNo, labl, lablLevel, formatType, ID, qstnLit, refs, min, max, universe, notes, txt, txtLevel
DATADSCRVARCATGRY	TxtLevel, name, catValu, labl, lablLevel, linkRefs, txt, missing, missType
DATADSCRVARINVALRNG	VALUE, name, ID
FILETXT	fileName, caseQnty, varQnty, logRecL, recPrCas, recNumTot, fileType, format, dataMsng
METHOD	timeMeth, dataCollector, frequenc, sampProc, collMode, resInstru, weight
SUMDSCR	StartDate, EndDate, nation, anlyUnit, universe, dataKind
TITLSTMT	titl, altTitl, IDNo, agency

A SAS datastep writes a SAS datastep

These datasets contain the information necessary to write a SAS program to read the data file documented by the DDI file. The SAS program to be output consists of a large comment at the beginning, containing metadata not needed for the mechanics of reading the file. Here is the beginning of that comment:

```
/*  SAS program to read ICPSR  6084
CBS News Monthly Poll #2, August 1992
August National Poll II, Republican National Convention

File Name:
    DS1: CBS News Monthly Poll #2, August 1992
Number of Cases:
    1,546
Number of Variables:
    70
Logical Record Length
    80
Records Per Case:
    3
```

The content of these elements needs to be edited to ensure that there won't be a "*/" printed in the middle of the comment. The DDI file for ICPSR 6084 also contained many tab characters.
The SAS macro used to edit the elements was:

```
        %macro putInCmnt(v=name, ve=vEdited, h=heading);
                &ve=compbl(tranwrd(translate(&v,' ','09'x),'*/','*_/'));
                if &ve ne ' ' then put / "&h"   / '     '  &ve;
        %mend putInCmnt;
```

The code to print the part of the beginning comment shown above was:

```
        set nostyles.titlstmt;
                titl= compbl(tranwrd(translate(titl,' ','09'x),'*/','*_/'));
                titl= compbl(tranwrd(translate(titl,' ','09'x),'*/','*_/'));
          put '/*  SAS program to read ' agency ' ' IDNo ;
                titl= compbl(tranwrd(translate(titl,' ','09'x),'*/','*_/'));
```

```
put titl;
        altTitl=compbl(tranwrd(translate(altTitl,' ','09'x),'*/','*_/'));
put altTitl;
        put //;
set Nostyles.Filetxt;
        %putInCmnt(v=fileName, ve=vEdited,  h=File Name:);
        %putInCmnt(v=caseQnty, ve=vEdited,  h=Number of Cases:);
        %putInCmnt(v=varQnty, ve=vEdited,  h=Number of Variables:);
        %putInCmnt(v=logRecl, ve=vEdited,  h=Logical Record Length);
        %putInCmnt(v=recPrCas, ve=vEdited,  h=Records Per Case:);
```

Since SAS format names are still restricted in length, the expedient thing to do was to construct format names like "V00015f" and then place a comment with the corresponding variable name. SAS missing values must also be assigned labels. The DDI elements *catValu* and *labl* were used to write value statements for a PROC FORMAT as in this example.

```
proc format;
value V00015f      /* format for variable Q1 */
1 ='Yes'
2 ='No'
.B ='DK/NA'
9 ='DK/NA'
```

In the example above the value "9" represents missing and will be replaced with the SAS missing value ".B" in the datastep.

Writing the DATA step to read the file described by the DDI file involves several SAS "DATA ._null_;" steps. One, shown below, writes the input statement, using the FILETXT, TITLSTMT, and a revised DATADSCRVAR datasets. Another writes a sequence of IF statements to substitute SAS unique missing values for the original missing values. Another writes variable label statements. A final "DATA _null_" statement assigns formats to variables.

```
data _null_;
                    /* begin writing datastep to read the variables */
  set DATADSCRVAR end=last;
  file reader lrecl=1024 mod;
  length ftype $ 3;
  if _n_=1 then do;
     set Nostyles.Filetxt;
     set Nostyles.titlstmt;
               put ///;
                                   /* remove unsafe characters */
    agency=translate(agency,'________','""";/*&%');
    IDNo=translate(IDNo,'________','""";/*&%');
    logrecl=translate(logrecl,'________','""";/*&%');
    RecSegNo=translate(RecSegNo,'________','""";/*&%');
    ftype=translate(ftype,'________','""";/*&%');
    StartPos=translate(StartPos,'________','""";/*&%');
    EndPos=translate(EndPos,'________','""";/*&%');

    put 'data ' agency +(-1) IDNo ';';
    put "infile  '&targetPath' LRECL=" logrecl    " PAD;";
    put "input ";
  end;
                                /*  position of each variable */
  if RecSegNo ne ' ' and StartPos ne ' ' and EndPos ne ' ' then do;
    if upcase(formatType) = 'NUMERIC' then ftype = '  ';
               else ftype = ' $ ';
    put "#" RecSegNo safename ftype StartPos +(-1) "-" EndPos ;
  end;

  if last then do;
    put ";";
  end;
run;
```

Here is an abbreviated copy of the SAS program written by the DDI processing SAS program.

```
/*  SAS program to read ICPSR  6084
CBS News Monthly Poll #2, August 1992
August National Poll II, Republican National Convention
File Name:
    DS1: CBS News Monthly Poll #2, August 1992
Number of Cases:
    1,546
Number of Variables:
    70
Logical Record Length
    80
Records Per Case:
    3
*/
proc format;
value V00015f      /* format for variable Q1 */
1 ='Yes'
2 ='No'
.B ='DK/NA'
9 ='DK/NA'
;
value V00067f      /* format for variable Q33a */
1 ='Under'
2 ='Over'
2 ='Won''t specify/Refused'
.I ='Won''t specify/Refused'
;
data ICPSR6084 ;
infile  'D:\data\icpsr\data\6084\da06084.txt' LRECL=80  PAD;
input
#1 cardno   1-1
#1 respno   2-6
#1 Q1   30-30
#2 cardno   1-1
#2 respno   2-6
#2 Q37   57-64
#3 cardno   1-1
#3 respno   2-6
#3 weight $ 49-54;
/* replace missing data with unique SAS missing values */
  if Q1  = 9  then Q1  = .B ;
  label Q1 ='Some people are registered to vote and others are not. Are you registered to vote in the precinct or election district where you live, or
aren''t you? ';
/*  This section will associate formats with each variable that has labeled categories */
/*  you may want to comment it out. */
  format Q1  V00015f.;
run;
```

Hand Debugging   ("Out out damn spot"?)

Unfortunately, having structured metadata doesn't always mean completely automated data input. In the case of our example ICPSR 6084 file, the SAS program written from the DDI file won't run without some hand debugging. In this case there are some variables (like Q33a) with values that have two different labels assigned. The format definition fails and then the datastep fails when trying to assign that format to a variable. Fortunately this is an easy error to fix – a lot easier than writing the whole SAS program from a printed codebook.

```
1684  value V00067f      /* format for variable Q33a */
1685  1 ='Under'
1686  2 ='Over'
1687  2 ='Won''t specify/Refused'
ERROR: This range is repeated, or values overlap: 2-2.
1688  .I ='Won''t specify/Refused'
1689  ;
```

The SAS program to read the DDI file and write a SAS program as well as the resultant SAS program will be available on the Web at …..