

Handling XML documents with SAS

Lex Jansen

New Jersey SAS Users Group

December 10, 2010



Unifying **Clinical Data** and **Submission Strategy** for the **Global Life Sciences Industry**

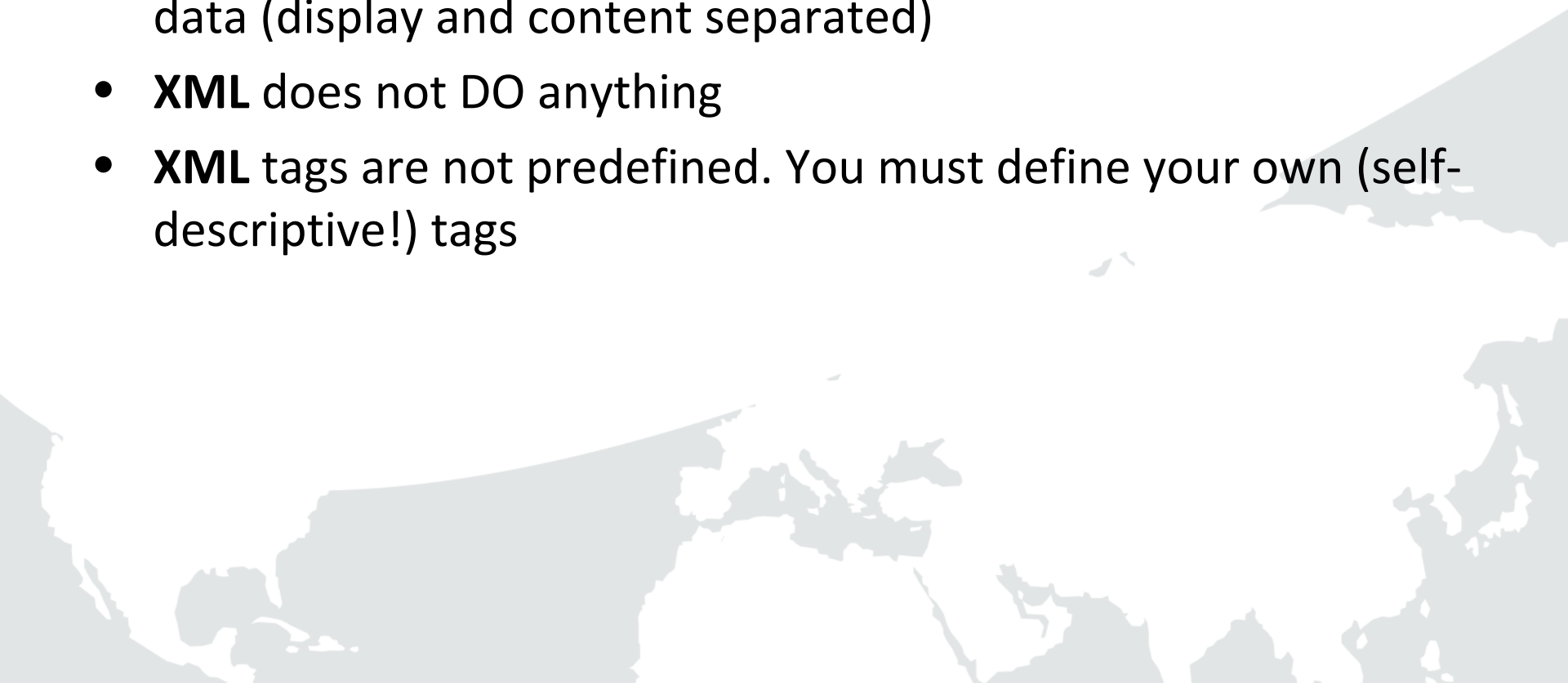
Agenda

- What is XML
- SAS and XML
 - SAS XML libname engine
 - SAS XML Mapper
- Pharmaceutical application: **define.xml**

What is XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<heartofrockandsoul>
  <entry rank="938">
    <artist>Gino Washington</artist>
    <title>Gino is a coward</title>
    <producer>Sonny Saunders</producer>
    <writer>Ronald Davis</writer>
    <label>Ric Tic</label>
    <year>1963</year>
    <billboard>Did not make pop charts</billboard>
  </entry>
</heartofrockandsoul>
```

What is XML

- **XML** stands for e**X**tensible **M**arkup **L**anguage
 - **XML** is a markup language much like HTML, with tags, elements and attributes
 - **XML** was designed to transport and store data, not to display data (display and content separated)
 - **XML** does not DO anything
 - **XML** tags are not predefined. You must define your own (self-descriptive!) tags
- 

What is XML

- Like HTML, **XML** makes use of *tags* (words bracketed by '<' and '>') and attributes (of the form name="value")
- BUT ... , HTML specifies what each tag and attribute means, and often how the text between them will look in a browser
- **XML** uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it
- In other words, if you see "<p>" in an XML file, do not assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person, (or maybe something that does not start with with a "p"?)

What is XML

There are 2 levels of correctness of an XML document:

- A **well-formed** XML document conforms to all of XML's syntax rules
- A **valid** XML document additionally conforms to certain semantic rules



What is XML

A **well-formed** XML file conforms to the rules of XML syntax:

- A single element (**root element**) contains all other elements in the document
- Elements have to be properly opened and closed
- Elements do not overlap, e.g. properly nested
- Attributes are properly quoted
- The document does not contain illegal characters.
Example: if the left opening bracket “<” is part of the content it should be substituted as “<”
- A conforming XML parser is not allowed to process an XML document that is not well-formed
- **Case matters!!!**

What is XML

root element

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<heartofrockandsoul>  
  <entry rank="265">  
    <artist>Nathaniel Mayer</artist>  
    <title>Village of love</title>  
    <producer>Producer not credited</producer>  
    <writer>Nathaniel Mayer & Devora Brown</writer>  
    <label>Fortune</label>  
    <year>1962</year>  
    <billboard>Did not make pop charts</billboard>  
  </entry>  
</heartofrockandsoul>
```

What is XML

- Predefined entities

| Character | Entity |
|-----------|--------|
| & | & |
| < | < |
| > | > |
| " | " |
| ' | ' |

Entities **&** and **<** **MUST** be used within the text value of an element or attribute

What is XML

An XML file is **valid** if it conforms to certain semantic rules:

- An **XML schema** is a description of these semantic rules
- A Schema defines constraints on the structure and contents of documents of that type
- A Schema defines allowed elements and attributes, order of elements, overall structure, etc ...
- A schema might describe that the content of a certain element, that contains a datetime value, is only valid if the value conforms to the ISO8601 standard
- A schema might describe that an attribute is only allowed a limited number of values

What is XML - XPath

- **XPath**: language for identifying particular parts of XML documents

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<heartofrockandsoul>
  <entry rank="938">
    <artist>Gino Washington</artist>
    <title>Gino is a coward</title>
    <producer>Sonny Saunders</producer>
    <writer>Ronald Davis</writer>
```

- Examples:

[/heartofrockandsoul/entry/artist](#) selects **artist** element

[/heartofrockandsoul/entry/@rank](#) selects **rank** attribute

Reading XML with SAS



The example XML file we use contains data from "THE HEART OF ROCK & SOUL, the 1001 Greatest Singles Ever Made"

by *Dave Marsh*.

See also:

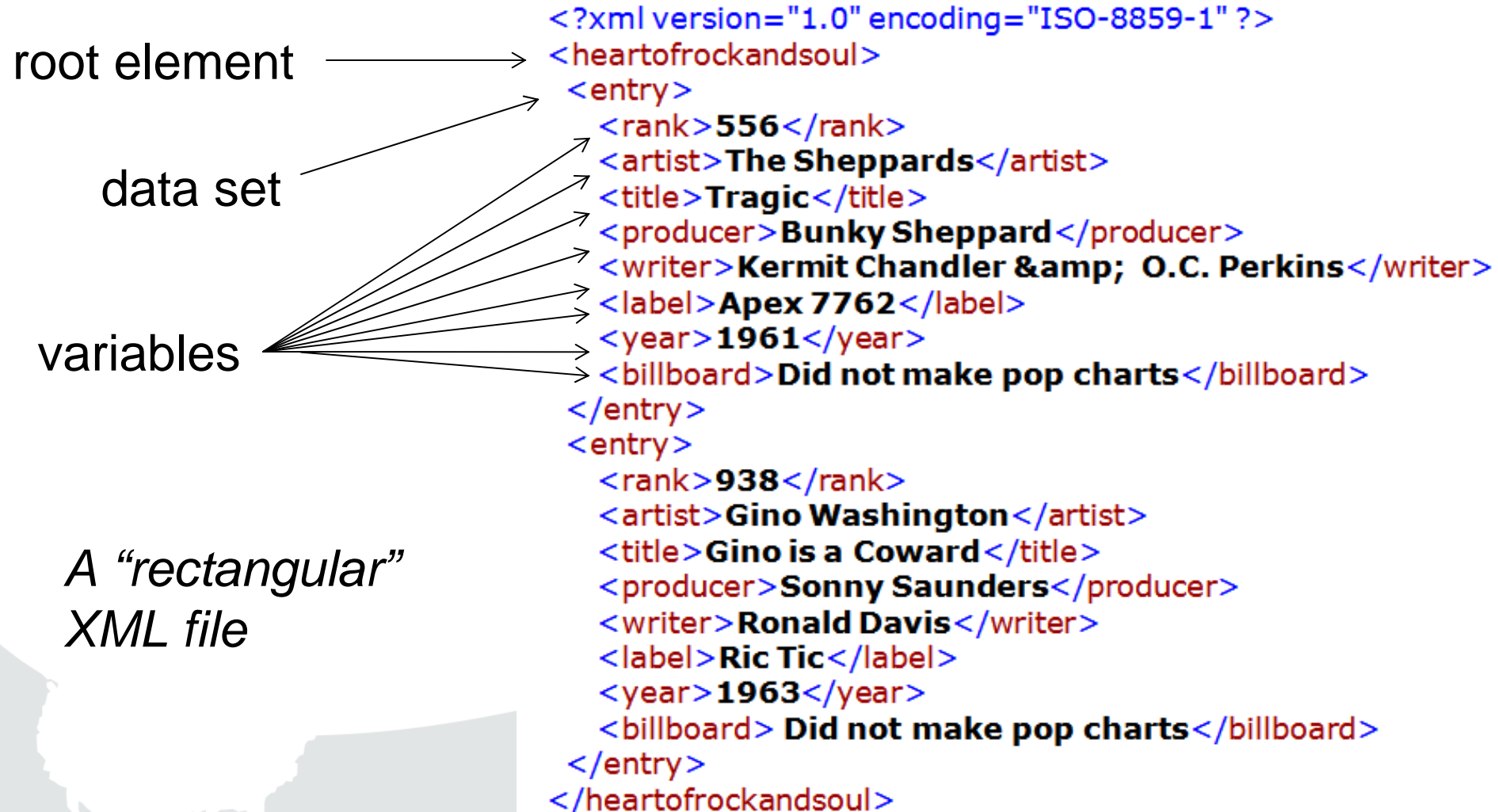
<http://www.lexjansen.com/marsh>



Reading XML files with SAS

- SAS can read generic XML files, that have the following characteristics:
 - The enclosing **root element** is comparable to a SAS **library**
 - A second-level element is translated to a **dataset** name
 - Other elements within that second level become SAS **variables**

Reading XML files with SAS



Reading XML files with SAS

```
* Exercise01-A_ReadXML.sas ;
```

```
FILENAME rocksoul "&WorkShop\xml\heartofrocknsoul-1.xml";
```

```
LIBNAME rs xml XMLFILEREFS=rocksoul;
```

```
PROC CONTENTS DATA=rs._ALL_ VARNUM;
```

```
RUN;
```

```
DATA work.RocknSoul1;
```

```
  SET rs.entry;
```

```
RUN;
```

```
PROC CONTENTS DATA=work.RocknSoul1 VARNUM;
```

```
RUN;
```

```
PROC PRINT DATA=work.RocknSoul1;
```

```
RUN;
```

Reading XML files with SAS

The CONTENTS Procedure

| | | | |
|---------------------|----------|----------------------|----|
| Data Set Name | RS.ENTRY | Observations | . |
| Member Type | DATA | Variables | 8 |
| Engine | XML | Indexes | 0 |
| Created | . | Observation Length | 0 |
| Last Modified | . | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | NO |
| Label | | | |
| Data Representation | Default | | |
| Encoding | Default | | |

Variables in Creation Order

| # | Variable | Type | Len | Format | Informat | Label |
|---|-----------|------|-----|--------|----------|-----------|
| 1 | BILLBOARD | Char | 23 | \$23. | \$23. | BILLBOARD |
| 2 | YEAR | Num | 8 | F8. | F8. | YEAR |
| 3 | LABEL | Char | 14 | \$14. | \$14. | LABEL |
| 4 | WRITER | Char | 44 | \$44. | \$44. | WRITER |
| 5 | PRODUCER | Char | 29 | \$29. | \$29. | PRODUCER |
| 6 | TITLE | Char | 36 | \$36. | \$36. | TITLE |
| 7 | ARTIST | Char | 28 | \$28. | \$28. | ARTIST |
| 8 | RANK | Num | 8 | F8. | F8. | RANK |

Reading XML files with SAS

& transformed to &

```
<entry>  
  <rank>556</rank>  
  <artist>The Sheppards</artist>  
  <title>Tragic</title>  
  <producer>Bunky Sheppard</producer>  
  <writer>Kermit Chandler & O.C. Perkins</writer>  
  <label>Apex 7762</label>  
  <year>1961</year>  
  <billboard>Did not make pop charts</billboard>  
</entry>
```

WRITER

```
Norman Whitfield & Barrett Strong  
Chuck Berry  
James Brown  
Brian Holland, Lamont Dozier & Edc  
Isaac Hayes & David Porter  
Nolan Strong and The Diablos  
Nathaniel Mayer & Devora Brown  
Dan Penn & Spooner Oldham  
Arlester (Dyke) Christian  
Chips Moman & Dan Penn  
Fred Parris  
Isaac Hayes & David Porter  
Kermit Chandler & O.C. Perkins  
Dickey Lee  
Ronald Davis
```

Reading XML files with SAS

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<heartofrockandsoul>
  <entry>
    <rank>556</rank>
    <artist>The Sheppards</artist>
    <title>Tragic</title>
    <producer>Bunky Sheppard</producer>
    <writer>Kermit Chandler & O.C. Perkins</writer>
    <label>Apex 7762</label>
    <year>1961</year>
    <billboard>Did not make pop charts</billboard>
  </entry>
  <entry>
    <rank>938</rank>
    <artist>Gino Washington</artist>
    <title>Gino is a Coward</title>
    <producer>Sonny Saunders</producer>
    <writer>Ronald Davis</writer>
    <label>Ric Tic</label>
    <year>1963</year>
    <billboard>Did not make pop charts</billboard>
  </entry>
</heartofrockandsoul>
```

Reading XML files with SAS

```
* Exercise01-B_ReadXML.sas ;
```

```
FILENAME rocksoul "&WorkShop\xml\heartofrocknsoul-2.xml";
```

```
LIBNAME rs xml XMLFILEREFS=rocksoul;
```

```
DATA work.RocknSoul2;
```

```
  SET rs.entry;
```

```
RUN;
```

```
PROC PRINT DATA=work.RocknSoul2;
```

```
RUN;
```

Reading XML files with SAS

```
29 * Excercise01-B_ReadXML.sas ;
30
31 FILENAME rock soul "&WorkShop\xml\heartofrocknsoul-2.xml";
32 LIBNAME rs xml XMLFILEREf=rock soul;
NOTE: Libref RS was successfully assigned as follows:
      Engine:          XML
      Physical Name:
33
34 DATA work.RocknSoul2;
35 SET rs.entry;
ERROR: There is an illegal character in the entity name.
       encountered during XMLMap parsing
       occurred at or near line 8, column 31
ERROR: XML describe error: Internal processing error.
36 RUN;

NOTE: The SAS System stopped processing this step because of err
WARNING: The data set WORK.ROCKNSOUL2 may be incomplete. When t
        observations and 0 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds
```

Reading XML files with SAS

- Characters like the ampersand (&) and the left angle bracket (<) must be escaped: **&** and **<**;
- To import an XML document that contains non-escaped characters, you can specify the **XMLPROCESS=RELAX** option on the LIBNAME.
- **Note: This is not recommended.**
If an XML document consists of non-escaped characters, the content is **not well-formed**.
- A conforming XML parser is not allowed to process an XML document that is not well-formed

Reading XML files with SAS

```
* Exercise01-C_ReadXML.sas ;
```

```
FILENAME rocksoul "&WorkShop\xml\heartofrocknsoul-2.xml";
```

```
LIBNAME rs xml XMLFILEREFF=rocksoul XMLPROCESS=RELAX;
```

```
DATA work.RocknSoul2;
```

```
  SET rs.entry;
```

```
RUN;
```

```
PROC PRINT DATA=work.RocknSoul2;
```

```
RUN;
```

Reading XML files with SAS

```
40 * Excercise01-C_ReadXML.sas ;
41
42 FILENAME rocksoul "&WorkShop\xml\heartofrocknsoul-2.xml";
43 LIBNAME rs xml XMLFILEREf=rocksoul XMLPROCESS=RELAX;
NOTE: Libref RS was successfully assigned as follows:
      Engine:          XML
      Physical Name:
44
45 DATA work.RocknSoul2;
46   SET rs.entry;
47 RUN;
```

```
NOTE: There were 16 observations read from the data set RS.ENTRY.
NOTE: The data set WORK.ROCKNSOUL2 has 16 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time          0.09 seconds
      cpu time           0.07 seconds
```

```
48
49 PROC PRINT DATA=work.RocknSoul2;
50 RUN;
```

```
NOTE: There were 16 observations read from the data set WORK.ROCKNSOUL2.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

Reading XML files with SAS

Let's make rank
an attribute

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<heartofrockandsoul>
  <entry>
    <rank>556</rank>
    <artist>The Sheppards</artist>
    <title>Tragic</title>
    <producer>Bunky Sheppard</producer>
    <writer>Kermit Chandler & O.C. Perkins</writer>
    <label>Apex 7762</label>
    <year>1961</year>
    <billboard>Did not make pop charts</billboard>
  </entry>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<heartofrockandsoul>
  <entry rank="556">
    <artist>The Sheppards</artist>
    <title>Tragic</title>
    <producer>Bunky Sheppard</producer>
    <writer>Kermit Chandler & O.C. Perkins</writer>
    <label>Apex 7762</label>
    <year>1961</year>
    <billboard>Did not make pop charts</billboard>
  </entry>
```

Reading XML files with SAS

```
* Exercise01-D_ReadXML.sas ;
```

```
FILENAME rocksoul "&WorkShop\xml\heartofrocknsoul-3.xml";
```

```
LIBNAME rs xml XMLFILEREFS=rocksoul;
```

```
PROC CONTENTS DATA=rs._ALL_ VARNUM;
```

```
RUN;
```

```
DATA work.RocknSoul3;
```

```
  SET rs.entry;
```

```
RUN;
```

```
PROC CONTENTS DATA=work.RocknSoul3 VARNUM;
```

```
RUN;
```

Reading XML files with SAS

The CONTENTS Procedure

| | | | |
|---------------------|----------|----------------------|----|
| Data Set Name | RS.ENTRY | Observations | . |
| Member Type | DATA | Variables | 7 |
| Engine | XML | Indexes | 0 |
| Created | . | Observation Length | 0 |
| Last Modified | . | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | NO |
| Label | | | |
| Data Representation | Default | | |
| Encoding | Default | | |

Variables in Creation Order

| # | Variable | Type | Len | Format | Informat | Label |
|---|-----------|------|-----|--------|----------|-----------|
| 1 | BILLBOARD | Char | 23 | \$23. | \$23. | BILLBOARD |
| 2 | YEAR | Num | 8 | F8. | F8. | YEAR |
| 3 | LABEL | Char | 14 | \$14. | \$14. | LABEL |
| 4 | WRITER | Char | 44 | \$44. | \$44. | WRITER |
| 5 | PRODUCER | Char | 29 | \$29. | \$29. | PRODUCER |
| 6 | TITLE | Char | 36 | \$36. | \$36. | TITLE |
| 7 | ARTIST | Char | 28 | \$28. | \$28. | ARTIST |

RANK ???

Reading XML files with SAS

- SAS can read generic XML files, that have the following characteristics:
 - The enclosing **root element** is comparable to a SAS **library**
 - A second-level element is translated to a **dataset** name
 - Other elements within that second level become SAS **variables**

**More complex XML hierarchy:
SAS needs a map to locate
elements and attributes**





Reading complex XML with the SAS XML Mapper

SAS XML Mapper

- XML files with a complex hierarchie need to be modeled to a **relational data model**
- Various tables (SAS datasets) that are related
- **SAS XML Mapper:**
 - tool that uses XPATH to create a **MAP** that describes how **XML** gets mapped to **SAS datasets**
 - free stand-alone Java client application available on the SAS product distribution disks

SAS XML Mapper

- Input for the **SAS XML Mapper**:
 - XML file
 - XML schema
- Before you start mapping you need to have an idea how your data model should look
- **SAS XML Mapper** does have an **AUTOMAP** feature, which may give you a **LOT** of SAS datasets that you still need to merge
- Or, you can map your datasets **manually** and have **control**

SAS XML Mapper

- Example: [define.xml](#)

Case Report Tabulation Data Definition Specification (define.xml)

Prepared by the
CDISC define.xml Team

Principal Editor: William Qubeck
Principal Contributors: Sally Cassells, Anthony Friebel, and the define.xml team

Notes to Readers

This version of the Case Report Tabulation Data Definition Specification supersedes all prior versions. Version 1.0.0 reflects changes from a comment period through the Health Level 7 (HL7) Regulated Clinical Research Information Management Technical Committee (RCRIM) in December 2003 (www.hl7.org) and CDISC's website in September 2004 as well as the work done by the define.xml team to add functionality, features, and additional documentation.

Version 1.0.0 incorporated the applicable comments, suggestions, and corrections received from the two comment periods specified above and is the official implementation version.

Revision History

| Date | Version | Summary of Changes | Primary Author |
|------------|---------|--|---|
| 2005-02-05 | 1.0.0 | This is the official implementation version of the Case Report Tabulation Data Definition Specification. | The define.xml team |
| 2005-02-09 | 1.0.0 | Administrative update. | Anthony Friebel, William Qubeck, Sally Cassells |

SAS XML Mapper

```
<ItemGroupDef
  OID="IG.MH" Name="MH" Repeating="Yes"
  IsReferenceData="No"
  SASDatasetName="MH"
  Purpose="Tabulation"
  def:Label="Medical History"
  def:Structure="One record per medical history event per subject"
  def:DomainKeys="STUDYID, USUBJID, MHCAT, MHTERM, MHSTDTC"
  def:Class="EVENTS"
  def:ArchiveLocationID="Location.MH">

  <ItemRef ItemOID="I.STUDYID"
    OrderNumber="1" Mandatory="Yes" Role="IDENTIFIER" />
  <ItemRef ItemOID="I.DOMAIN"
    OrderNumber="2" Mandatory="Yes" Role="IDENTIFIER" />
  <ItemRef ItemOID="I.USUBJID"
    OrderNumber="3" Mandatory="Yes" Role="IDENTIFIER" />
  ...
  <ItemRef ItemOID="I.MHTERM"
    OrderNumber="8" Mandatory="Yes" Role="TOPIC" />
  ...

  <def:leaf ID="Location.MH" xlink:href="mh.xpt">
    <def:title>mh.xpt</def:title>
  </def:leaf>
</ItemGroupDef>
```

```
<ItemRef ItemOID="I.VSTESTCD" OrderNumber="6" Mandatory="Yes"  
  Role="TOPIC" RoleCodeListOID="RoleCodeList" />
```

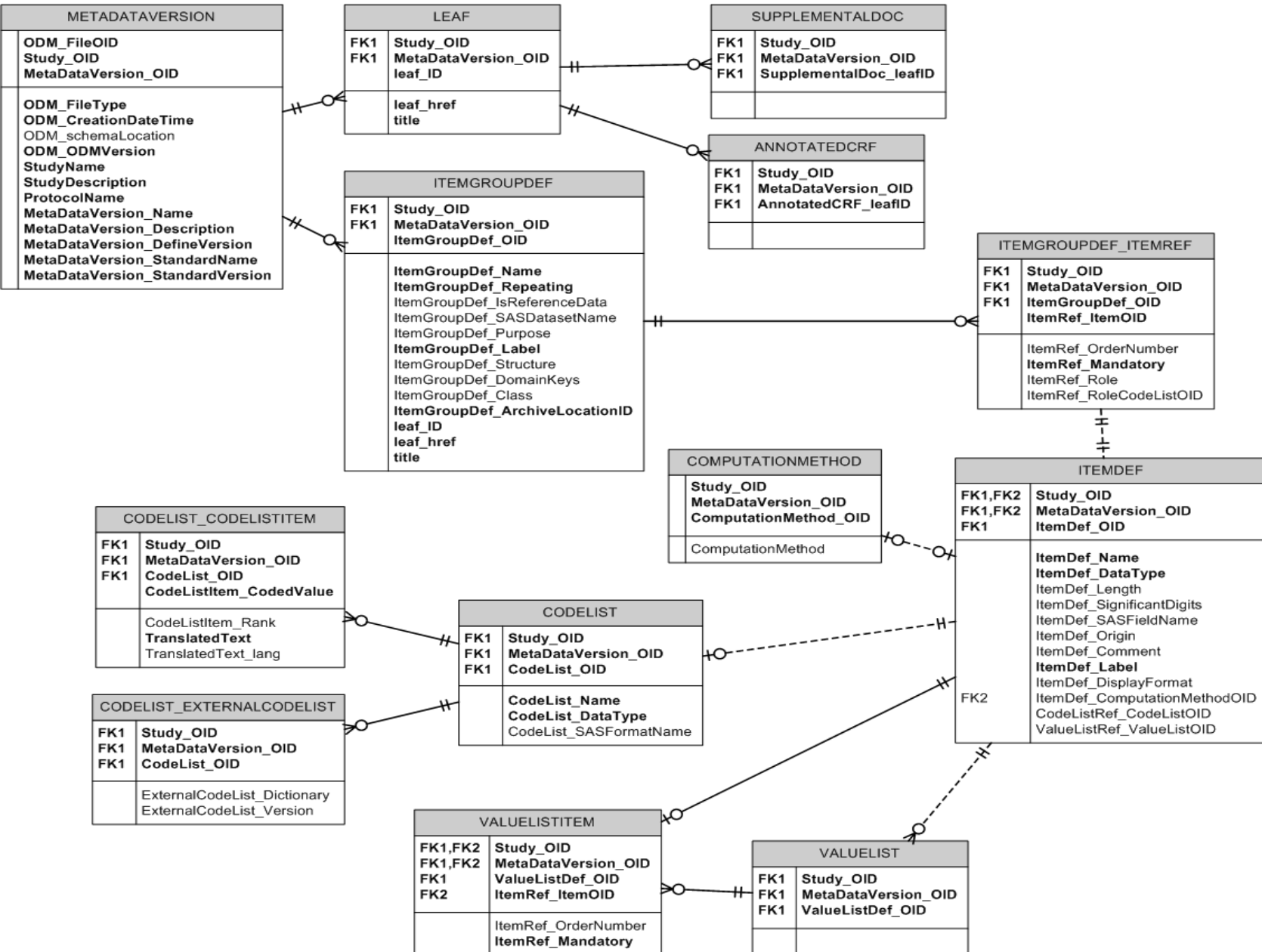
```
<ItemRef ItemOID="I.VSBLFL" OrderNumber="14" Mandatory="No"  
  Role="RECORD QUALIFIER" RoleCodeListOID="RoleCodeList" />
```

```
<ItemDef OID="I.VSTESTCD"  
  Name="VSTESTCD" DataType="text"  
  Length="8" Origin="CRF Page 9"  
  def:Label="Vital Signs Test Short Name">  
    <def:ValueListRef ValueListOID="VL.VSTESTCD" />  
</ItemDef>
```

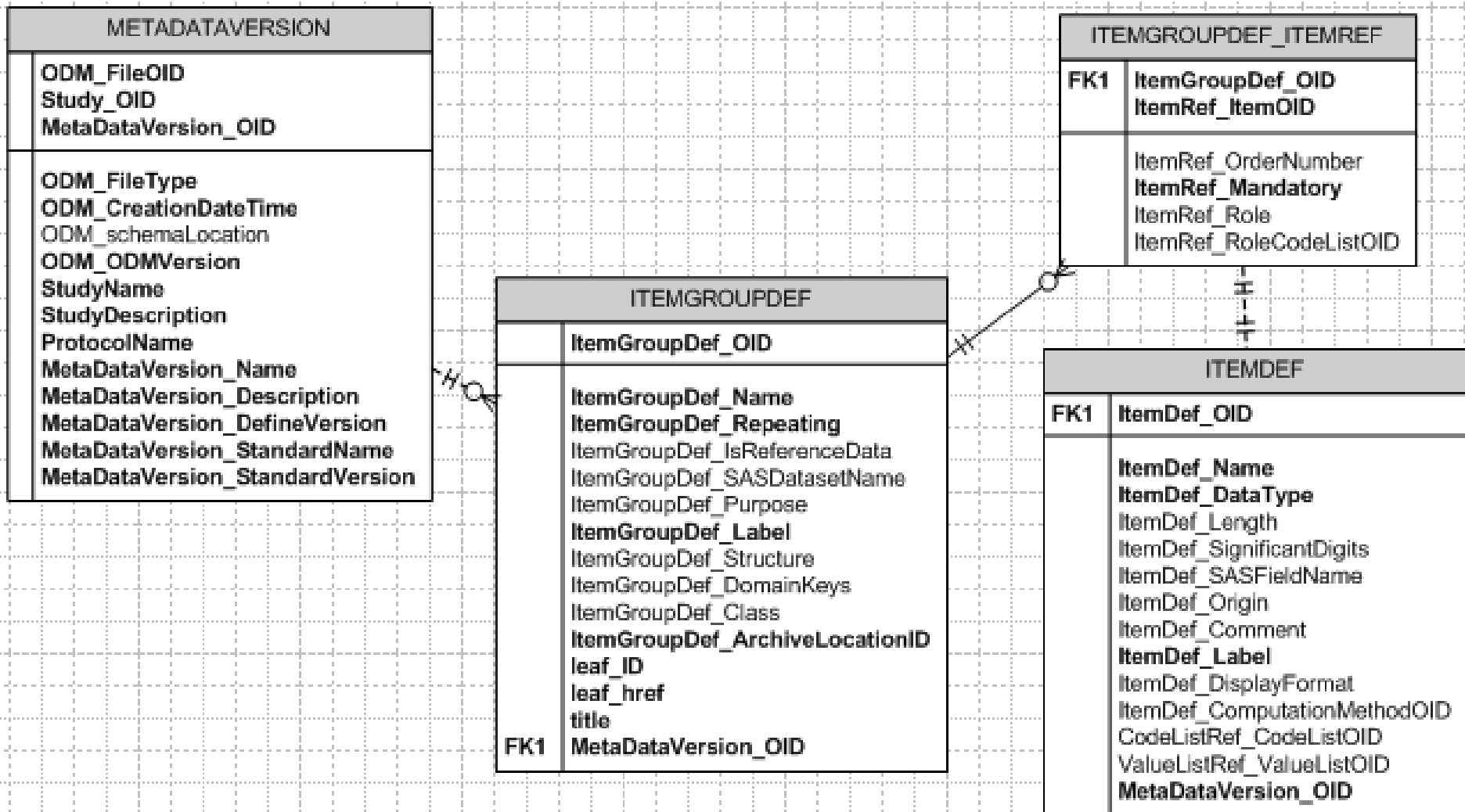
...

```
<ItemDef OID="I.VSBLFL" Name="VSBLFL" DataType="text" Length="1"  
  Origin="DERIVED" Comment="Derivation of baseline: Safety patients only: VSBLFL=Y  
  for last non missing record on or before the first dose date (RFSTDTC)."  
  def:Label="Baseline Flag"  
  def:ComputationMethodOID="CM.VSBLFL">  
    <CodeListRef CodeListOID="CL.NY" />  
</ItemDef>
```

define.xml



define.xml





Condensed Full Schema

- MetaDataVersion [1] {1}
 - Attributes [1] {1}
 - def.AnnotatedCRF [1] {1}
 - def.leaf [1] {1}
 - def.ComputationMethod [2] {2}
 - def.ValueListDef [19] {19}
 - ItemGroupDef [28] {28}
 - ItemDef [389] {389}**
 - Attributes [1] {389}
 - OID [1] {389}
 - Name [1] {389}
 - DataType [1] {389}
 - Length [1] {389}
 - Origin [1] {389}
 - Comment [1] {107}
 - def.Label [1] {389}
 - SignificantDigits [1] {14}
 - def.ComputationMethodOID [1] {2}
 - CodeListRef [1] {62}
 - Attributes [1] {62}
 - CodeListOID [1] {62}
 - def.ValueListRef [1] {19}
 - Attributes [1] {19}
 - ValueListOID [1] {19}
 - CodeList [29] {29}

define.xml structure

Properties Format Condition Enumeration Class XMLMap Settings Output

Name: ItemDef_OID
 Description: OID
 Path: /ODM/Study/MetaDataVersion/ItemDef/@OID
 End Path: Begin/End
 Retain Replace

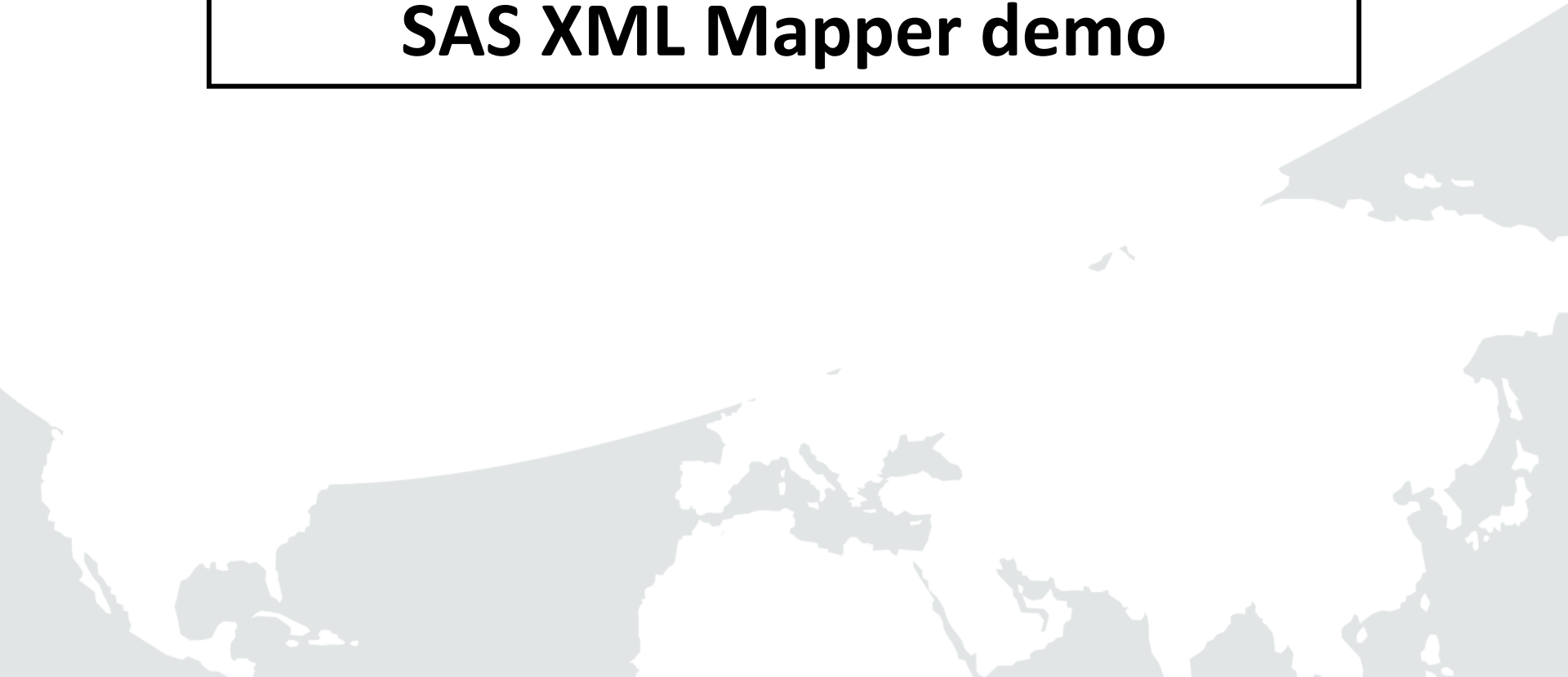
- ItemGroupDef
 - ItemGroupDef_ItemRef
 - ItemDef
 - Study_OID
 - MetaDataVersion_OID
 - ItemDef_OID**
 - ItemDef_Name
 - ItemDef_DataType
 - ItemDef_Length
 - ItemDef_SignificantDigits
 - ItemDef_SASFieldName
 - ItemDef_Origin
 - ItemDef_Comment
 - ItemDef_Label
 - ItemDef_DisplayFormat
 - ItemDef_ComputationMethodOID
 - CodeListRef_CodeListOID
 - ValueListRef_ValueListOID

SAS data set Metadata

Table: ItemDef Row: 24 / 389 Columns: 6 / 15 SAS formats and informats are not applied to this view.

| | ItemDef_OID | ItemDef_Name | ItemDef_D... | ItemDef_Le... | ItemDef_S... | ItemDef... | ItemDef_Origin | ItemDef_Comment |
|----|----------------------|--------------|--------------|---------------|--------------|------------|-----------------|---|
| 16 | SUPPVS.QNAM.VSCSI... | VSCSIND | text | 2 | | | DERIVED | Only created if value qualifies as pote |
| 17 | AE.AESEQ | AESEQ | integer | 1 | | | DERIVED | Sequential number uniquely identifiyir |
| 18 | AE.AESPID | AESPID | text | 2 | | | SPONSOR DEFINED | ID of original SAS dataset |
| 19 | AE.AETERM | AETERM | text | 25 | | | CRF Page 34 | |
| 20 | AE.AEMODIFY | AEMODIFY | text | 9 | | | SPONSOR DEFINED | |
| 21 | AE.AEDECOD | AEDECOD | text | 18 | | | DERIVED | MedDRA version 8.0 |
| 22 | AE.AEBODSYS | AEBODSYS | text | 52 | | | DERIVED | MedDRA version 8.0 |
| 23 | AE.AESEV | AESEV | text | 8 | | | CRF Page 34 | |

SAS XML Mapper demo





Lex Jansen

Senior Consultant, Clinical Data Strategies

ljansen@octagonresearch.com