Accessing the Metadata from Define-XML

Lex Jansen

Principal Software Developer @ SAS

PharmaSUG 2018

Paper SS11

Lex Jansen

- 16 years in an IT/Standards role in Biostatistics dept. at Organon
- 4 years consultant to help companies implement CDISC
- Last 7 years as a Principle Software Developer at SAS working on SAS Clinical Standards Toolkit (implementing mostly XML based standards (Define-XML, ODM, Dataset-XML)) and SAS Life Science Analytics Framework

- Core member of the CDISC XML Technologies team since 2008.
- Core member of the CDISC Define-XML development team.
  - One of the main **Define-XML v2** developers.
  - Developer of CDISC **Define-XML v2 stylesheet**.
- Core member of the CDISC **Dataset-XML** development team.
- Core member of ADaM Metadata team
  - One of the main developers of the **Analysis Results Metadata** v1.0 for Define-XML v2.0 extension

Agenda

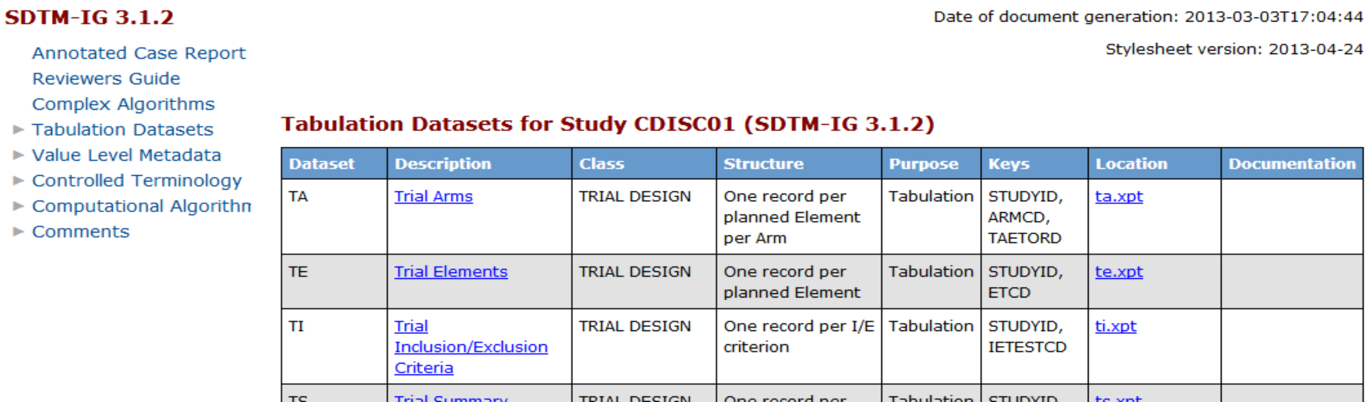
- Introduction
- Metadata in Define-XML
- SAS XML LIBNAME engine and SAS XML Mapper
- Slurping XML with PROC GROOVY
- Transforming XML with PROC XSL
- SAS Clinical Standards Toolkit

Introduction

Accessing the Metadata from Define-XML

Define-XML v2.0

- CDISC XML Technology standard.
- Provides machine readable metadata for SEND, SDTM, ADaM, or legacy data sets.
- Provides a table of contents for CDISC datasets.
- Based on the CDISC Operational Data Model (ODM)
- Required by FDA (USA) and PMDA (Japan) for all CDISC submissions.



SDTM-IG 3.1.2

Date of document generation: 2013-03-03T17:04:44

Stylesheet version: 2013-04-24

Tabulation Datasets for Study CDISC01 (SDTM-IG 3.1.2)

Dataset	Description	Class	Structure	Purpose	Keys	Location	Documentation
TA	Trial Arms	TRIAL DESIGN	One record per planned Element per Arm	Tabulation	STUDYID, ARMCD, TAETORD	ta.xpt	
TE	Trial Elements	TRIAL DESIGN	One record per planned Element	Tabulation	STUDYID, ETCD	te.xpt	
TI	Trial Inclusion/Exclusion Criteria	TRIAL DESIGN	One record per I/E criterion	Tabulation	STUDYID, IETESTCD	ti.xpt	
TS	Trial Summary	TRIAL DESIGN	One record per	Tabulation	STUDYID	ts.xpt	

Accessing the Metadata from Define-XML

Why do this?

- Define-XML is not just for viewing in a browser!
- The metadata in Define-XML can be used to drive **automation**
- Define-XML can hold dataset specifications
- Create 0-observation datasets for the metadata in Define-XML
- Use study metadata from a Define-XML document as a starting point for the metadata in a similar study

Metadata in Define-XML

Accessing the Metadata from Define-XML

Metadata in Define-XML

- **Study**
  - Study Name, Study Description, Protocol name
- **Datasets**
  - Name, Label, Domain, Structure, Class, Purpose, Keys, Comments, Documentation, Referenced Documents, Dataset Location, ...
- **Variables**
  - Name, Label, Data Type, Length, Significant Digits, Display Format, Controlled Terminology, Origin, Derivations, Comments, Documentation, Referenced Documents, ...
- **Controlled Terminology / Dictionaries**
- **Derivations** (algorithms, computations, methods)
- **Comments**, including referenced documents
- Supporting **Documents** (aCRF, Supplemental Data Definitions, Reviewer Guides, ...)
- (Parameter) **Value Level Metadata**
- **Analysis Results Metada**

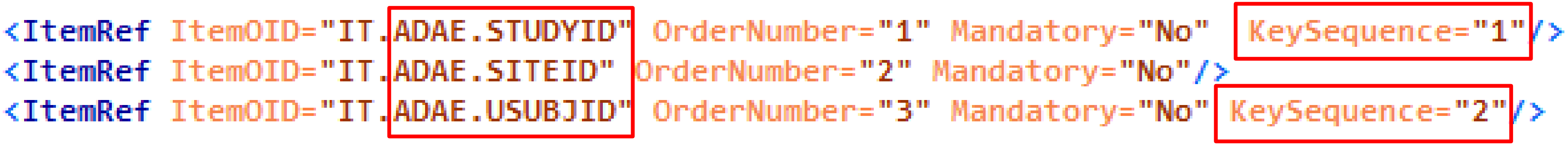
Accessing the Metadata from Define-XML

Unique Object Identfiers in Define-XML

- In Define-XML, there are many instances where one object needs to **reference** another object
- Elements are given a unique **identifier** (OID), and use that OID to reference the target element
- Values used for OIDs can follow any convention, or could be randomly generated
- The **only allowed use of OIDs** is to define an unambiguous **link** between a definition of an object and references to it.
- **No meaning should be attached to the value an OID.**

Accessing metadata from Define-XML

Example: the list of key variables in a datasets



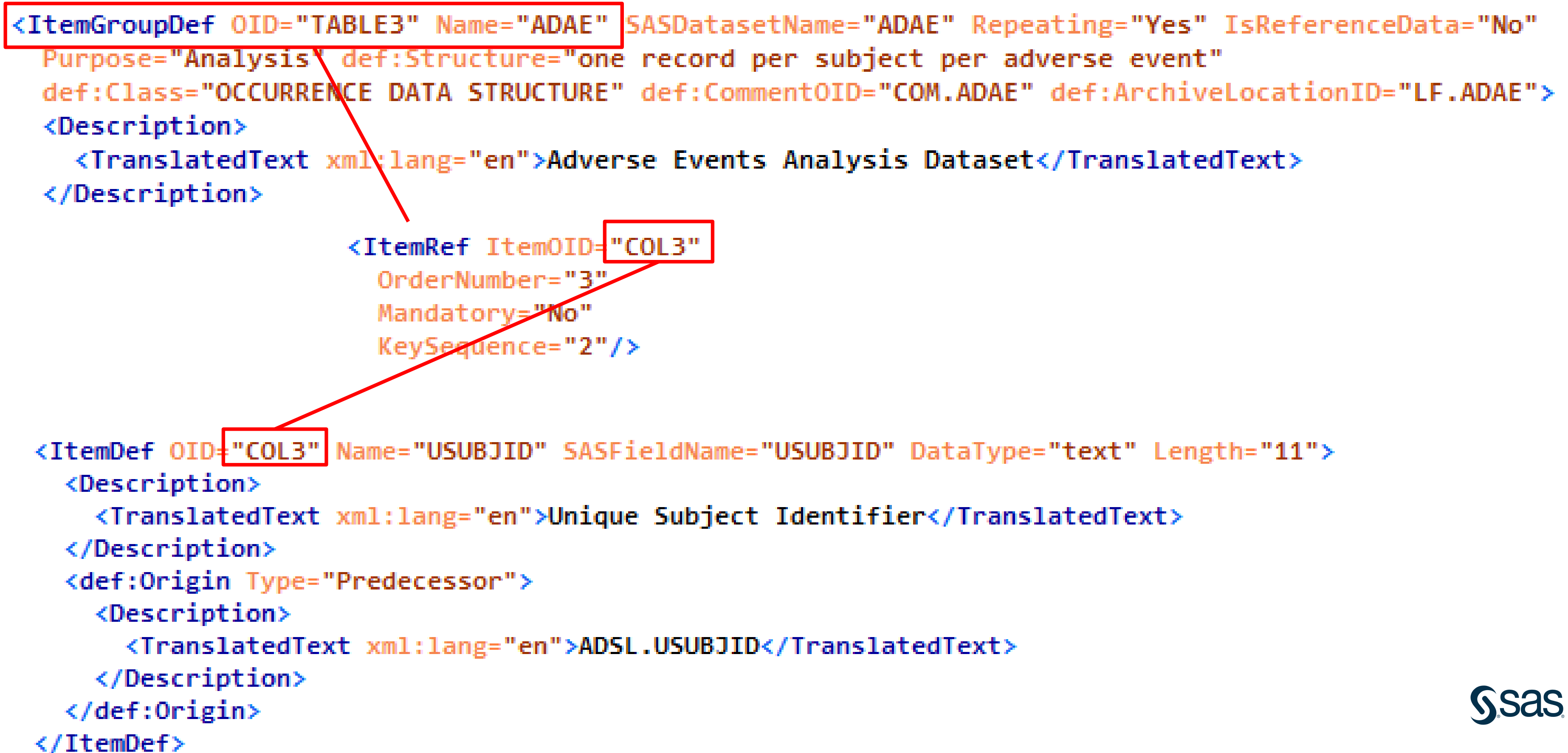
```
<ItemRef ItemOID="IT.ADAE.STUDYID" OrderNumber="1" Mandatory="No"  KeySequence="1"/>
<ItemRef ItemOID="IT.ADAE.SITEID" OrderNumber="2" Mandatory="No"/>
<ItemRef ItemOID="IT.ADAE.USUBJID" OrderNumber="3" Mandatory="No" KeySequence="2"/>
```

That is easy! Read the XML with a SAS DATA Step and a Regular Expression that looks for “**KeySequence=**” and “**<ItemRef**“ and we can extract the SAS data set name and variable name.

Hmmmm ... maybe not .... this is a **not a good practice**.

Accessing metadata from Define-XML

Example: the list of key variables in a dataset



```xml
<ItemGroupDef OID="TABLE3" Name="ADAE" SASDatasetName="ADAE" Repeating="Yes" IsReferenceData="No"
  Purpose="Analysis" def:Structure="one record per subject per adverse event"
  def:Class="OCCURRENCE DATA STRUCTURE" def:CommentOID="COM.ADAE" def:ArchiveLocationID="LF.ADAE">
  <Description>
    <TranslatedText xml:lang="en">Adverse Events Analysis Dataset</TranslatedText>
  </Description>

                        <ItemRef ItemOID="COL3"
                          OrderNumber="3"
                          Mandatory="No"
                          KeySequence="2"/>


 <ItemDef OID="COL3" Name="USUBJID" SASFieldName="USUBJID" DataType="text" Length="11">
   <Description>
     <TranslatedText xml:lang="en">Unique Subject Identifier</TranslatedText>
   </Description>
   <def:Origin Type="Predecessor">
     <Description>
       <TranslatedText xml:lang="en">ADSL.USUBJID</TranslatedText>
     </Description>
   </def:Origin>
 </ItemDef>
```

Accessing the Metadata from Define-XML

Metadata in Define-XML

In this presentation we will focus on dataset and variable metadata:

- Datasets
  - Name, Label, Domain, Structure, Class, Purpose, Keys, Comments, Documentation, Referenced Documents, Dataset Location, ...
- Variables
  - Name, Label, Data Type, Length, Significant Digits, Display Format, Controlled Terminology, Origin, Derivations, Comments, Documentation, Referenced Documents, ...

Accessing the Metadata from Define-XML

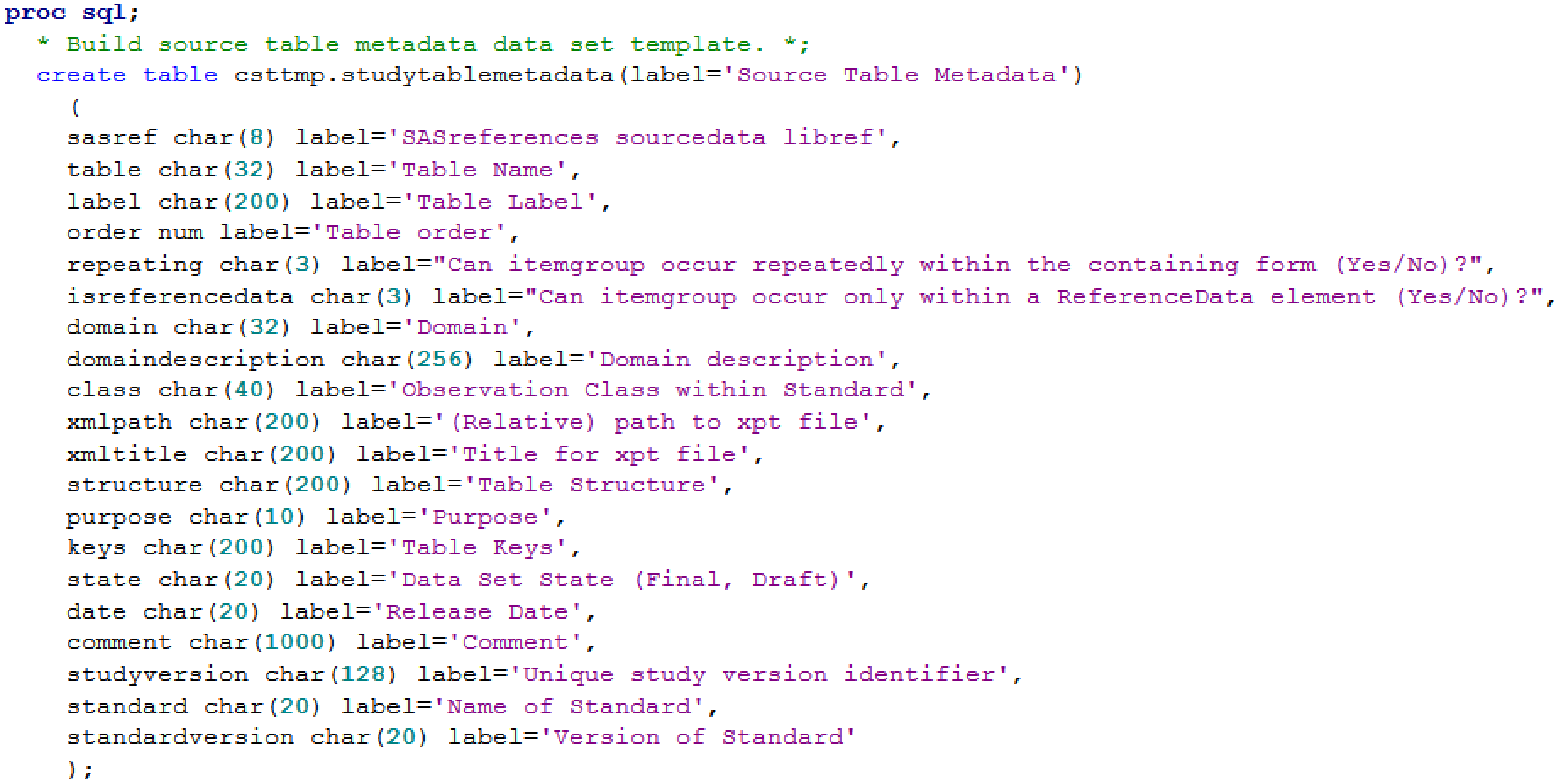
We will look at 4 ways to extract the metadata from Define-XML. These are robust and **XML aware** technologies.

- The **SAS XML LIBNAME Engine** and the **SAS XML MAPPER**
- Slurping XML with **PROC GROOVY**
- Transforming XML with **PROC XSL**
- **SAS Clinical Standards Toolkit**

All code will be available after the conference at lexjansen.com

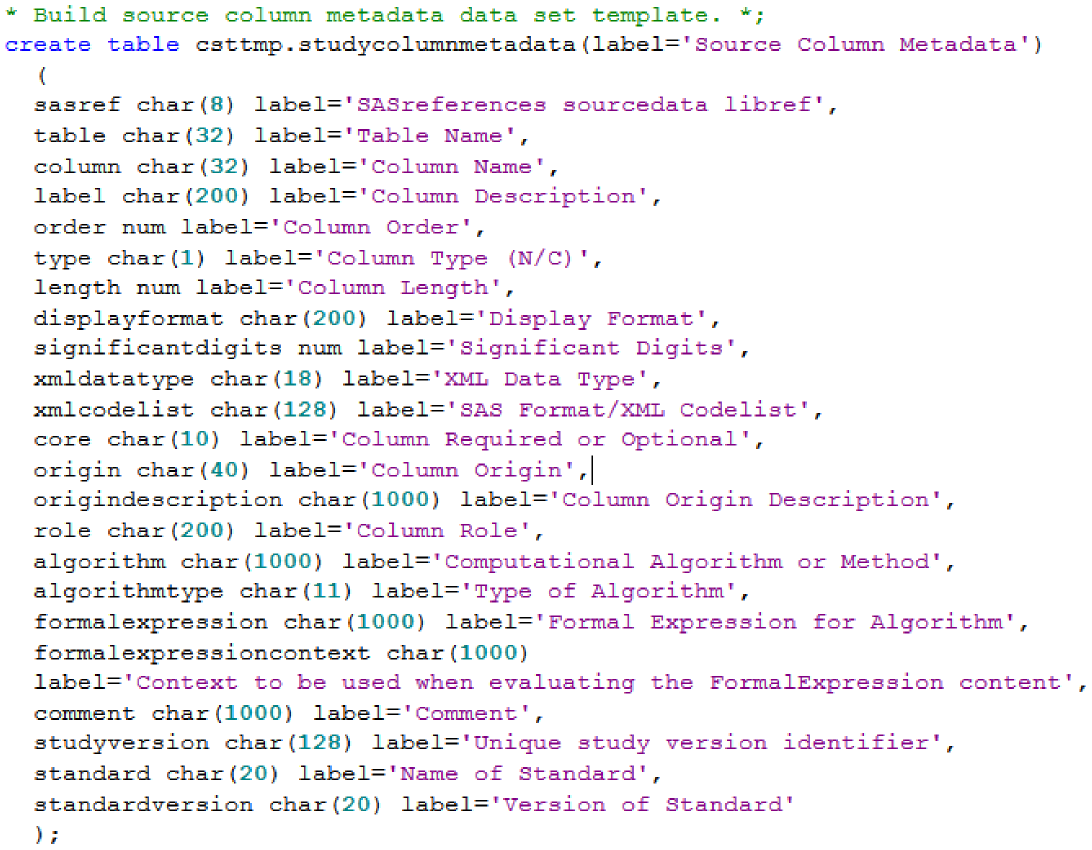
We will use templates to define the target datasets to create.

Accessing the Metadata from Define-XML



```
proc sql;
  * Build source table metadata data set template. *;
  create table csttmp.studytablemetadata(label='Source Table Metadata')
    (
    sasref char(8) label='SASreferences sourcedata libref',
    table char(32) label='Table Name',
    label char(200) label='Table Label',
    order num label='Table order',
    repeating char(3) label="Can itemgroup occur repeatedly within the containing form (Yes/No)?",
    isreferencedata char(3) label="Can itemgroup occur only within a ReferenceData element (Yes/No)?",
    domain char(32) label='Domain',
    domaindescription char(256) label='Domain description',
    class char(40) label='Observation Class within Standard',
    xmlpath char(200) label='(Relative) path to xpt file',
    xmltitle char(200) label='Title for xpt file',
    structure char(200) label='Table Structure',
    purpose char(10) label='Purpose',
    keys char(200) label='Table Keys',
    state char(20) label='Data Set State (Final, Draft)',
    date char(20) label='Release Date',
    comment char(1000) label='Comment',
    studyversion char(128) label='Unique study version identifier',
    standard char(20) label='Name of Standard',
    standardversion char(20) label='Version of Standard'
    );
```

Accessing the Metadata from Define-XML



```
* Build source column metadata data set template. *;
create table csttmp.studycolumnmetadata(label='Source Column Metadata')
  (
  sasref char(8) label='SASreferences sourcedata libref',
  table char(32) label='Table Name',
  column char(32) label='Column Name',
  label char(200) label='Column Description',
  order num label='Column Order',
  type char(1) label='Column Type (N/C)',
  length num label='Column Length',
  displayformat char(200) label='Display Format',
  significantdigits num label='Significant Digits',
  xmldatatype char(18) label='XML Data Type',
  xmlcodelist char(128) label='SAS Format/XML Codelist',
  core char(10) label='Column Required or Optional',
  origin char(40) label='Column Origin',
  origindescription char(1000) label='Column Origin Description',
  role char(200) label='Column Role',
  algorithm char(1000) label='Computational Algorithm or Method',
  algorithmtype char(11) label='Type of Algorithm',
  formalexpression char(1000) label='Formal Expression for Algorithm',
  formalexpressioncontext char(1000)
  label='Context to be used when evaluating the FormalExpression content',
  comment char(1000) label='Comment',
  studyversion char(128) label='Unique study version identifier',
  standard char(20) label='Name of Standard',
  standardversion char(20) label='Version of Standard'
  );
```

SAS XML LIBNAME engine and SAS XML Mapper

SAS XML LIBNAME engine and SAS XML Mapper

- SAS can read and write generic XML files with the XML LIBNAME engine, 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

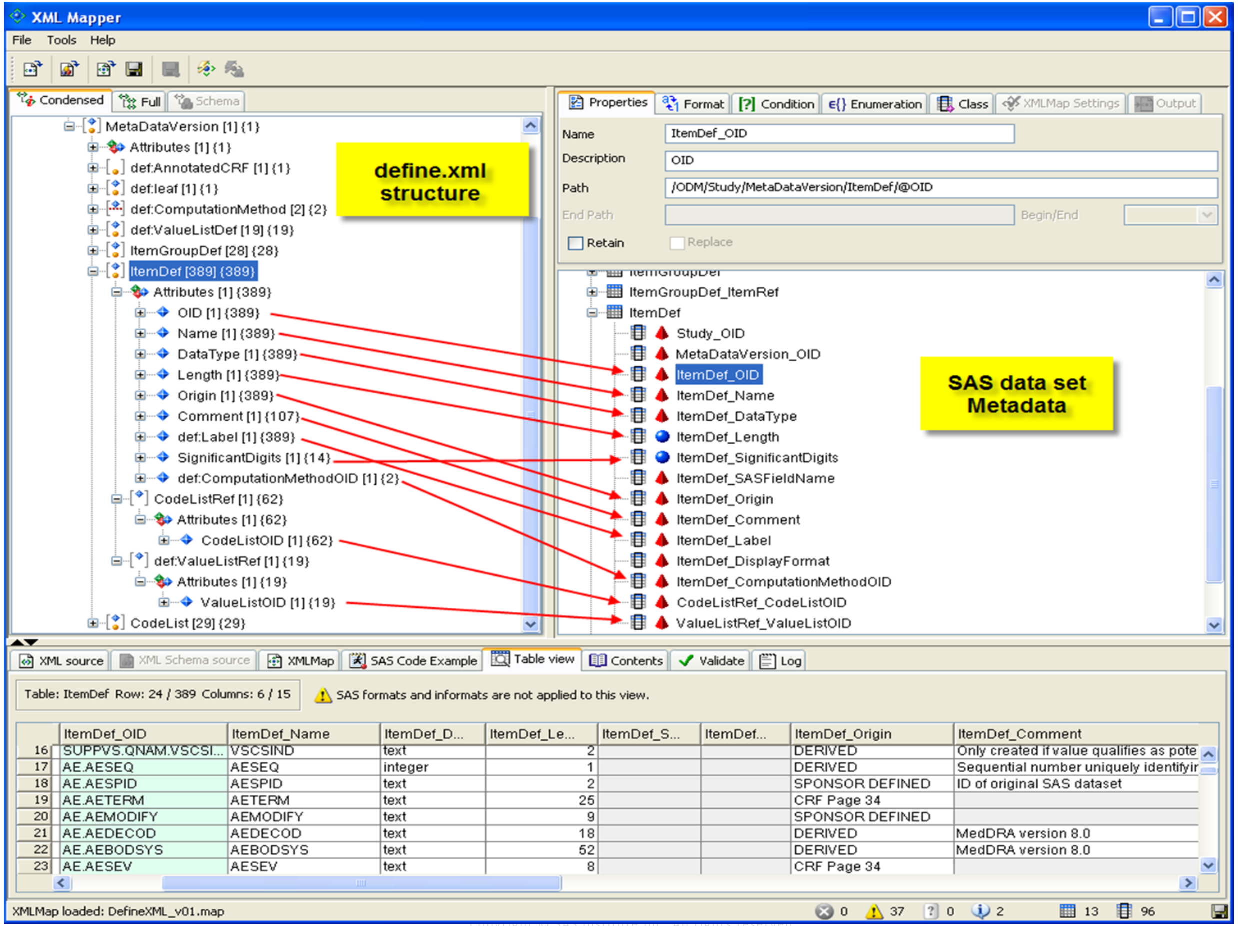
- More complex XML hierarchy: SAS needs a **map** to locate elements and attributes

SAS XML LIBNAME engine and SAS XML Mapper

- Complex XML files like Define-XML need to be modeled to a **relational data model**

- **SAS XML Mapper**:
  - Free stand-alone Java tool that uses **XPATH** to create an **XMLMap** that describes how XML gets mapped to SAS datasets
  - Available for download at the SAS Support site
  - Before you start mapping you need to have an idea how your data model should look

SAS XML LIBNAME engine and SAS XML Mapper



define.xml structure

SAS data set Metadata

Name: ItemDef_OID

Description: OID

Path: /ODM/Study/MetaDataVersion/ItemDef/@OID

End Path:

Begin/End:

□ Retain □ Replace

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 identifyir
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	

XMLMap loaded: DefineXML_v01.map   0   37   0   2   13   96

SAS XML LIBNAME engine and SAS XML Mapper

Auto Mapping

- **SAS XML Mapper** does have an **AUTOMAP** feature, which will give you a **LOT** of SAS datasets that you need to merge
- Or, you can map your datasets **manually** and have **control** (2010, http://support.sas.com/resources/papers/proceedings10/157-2010.pdf)
- Possible inputs for the **SAS XML Mapper**:
  - A specific instance of the Define-XML document
  - The XML schema that describes the Define-XML documents

SAS XML LIBNAME engine and SAS XML Mapper

Auto Mapping

- Read Thomas Cox's paper:
  Advanced XML Processing with SAS® 9.3
  http://support.sas.com/resources/papers/proceedings12/220-2012.pdf
- **Automapping** functionality in SAS XML Mapper was introduced in SAS 9.1.3 SP3
- Never intended to fully replace the drag-and-drop creation of XMLMaps
- Why does the **automapper** create so many tables?
  - Automapping algorithm creates a representation of the XML that is as relational as possible from the available context
  - XML file instances might not fully represent the cardinality of all analogous XML files

SAS XML LIBNAME engine and SAS XML Mapper

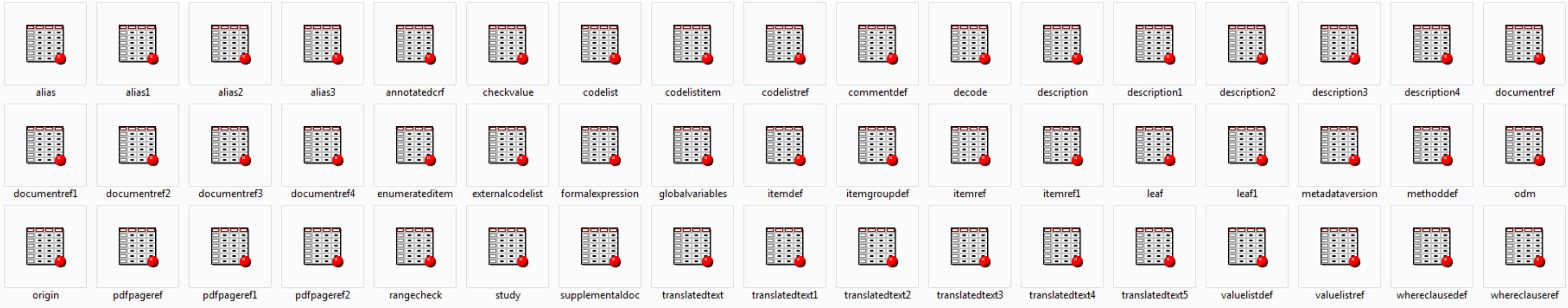
Strategy in this paper

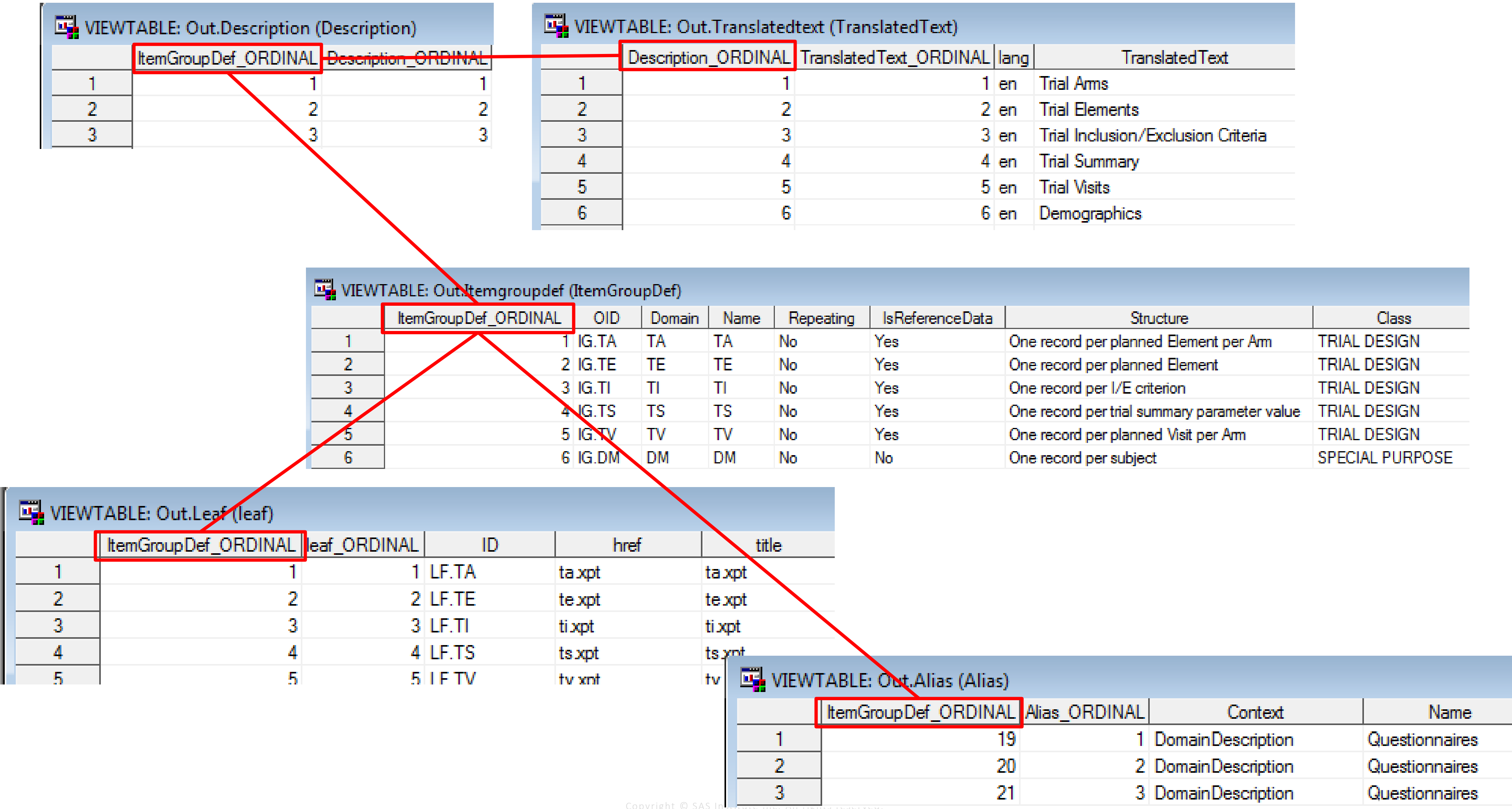
- Use automapping on an instance of a Define-XML document that contained all elements and attributes to be expected
- No efforts were made to tweak the lengths of the variables to be created. You may need to do this, to be able to re-use the XMLMap
- The final variable lengths were set by using the metadata templates that were mentioned earlier in this paper.
- Some datatypes needed to be tweaked.
- Use **automap=reuse** – create an XMLMap if it does not exist, otherwise use it. This would allow to tweak after the first run.

Needs a good understanding of the resulting datasets to be able to merge.

SAS XML LIBNAME engine and SAS XML Mapper

```
%let Root=C:/_Data/Presentations/PharmaSUG_2018/Accessing_DefineXML;

libname tmplts "&Root/templates";
libname out "&Root/xmlmap/out";

filename define "&Root/xml/define2-0-0-example-sdtm.xml";
filename map "&Root/xmlmap/definexml_auto.map";
libname define xmlv2 automap=reuse xmlmap=map prefixattributes=no;

proc copy in=define out=out;
run;
```





VIEWTABLE: Out.Description (Description)

	ItemGroupDef_ORDINAL	Description_ORDINAL
1	1	1
2	2	2
3	3	3

VIEWTABLE: Out.Translatedtext (TranslatedText)

	Description_ORDINAL	TranslatedText_ORDINAL	lang	TranslatedText
1	1	1	en	Trial Arms
2	2	2	en	Trial Elements
3	3	3	en	Trial Inclusion/Exclusion Criteria
4	4	4	en	Trial Summary
5	5	5	en	Trial Visits
6	6	6	en	Demographics

VIEWTABLE: Out.Itemgroupdef (ItemGroupDef)

	ItemGroupDef_ORDINAL	OID	Domain	Name	Repeating	IsReferenceData	Structure	Class
1	1	IG.TA	TA	TA	No	Yes	One record per planned Element per Arm	TRIAL DESIGN
2	2	IG.TE	TE	TE	No	Yes	One record per planned Element	TRIAL DESIGN
3	3	IG.TI	TI	TI	No	Yes	One record per I/E criterion	TRIAL DESIGN
4	4	IG.TS	TS	TS	No	Yes	One record per trial summary parameter value	TRIAL DESIGN
5	5	IG.TV	TV	TV	No	Yes	One record per planned Visit per Arm	TRIAL DESIGN
6	6	IG.DM	DM	DM	No	No	One record per subject	SPECIAL PURPOSE

VIEWTABLE: Out.Leaf (leaf)

	ItemGroupDef_ORDINAL	leaf_ORDINAL	ID	href	title
1	1	1	LF.TA	ta.xpt	ta.xpt
2	2	2	LF.TE	te.xpt	te.xpt
3	3	3	LF.TI	ti.xpt	ti.xpt
4	4	4	LF.TS	ts.xpt	ts.xpt
5	5	5	LF.TV	tv.xpt	tv

VIEWTABLE: Out.Alias (Alias)

	ItemGroupDef_ORDINAL	Alias_ORDINAL	Context	Name
1	19	1	DomainDescription	Questionnaires
2	20	2	DomainDescription	Questionnaires
3	21	3	DomainDescription	Questionnaires

Slurping XML with PROC GROOVY

Accessing the Metadata from Define-XML

Groovy

- Groovy is a dynamic language that runs on the Java Virtual Machine (JVM)
- PROC GROOVY enables SAS code to execute Groovy code on the JVM
- PROC GROOVY can run Groovy statements
  - that are part of your SAS code
  - that are in files that you specify on the PROC GROOVY statement
  - or parse Groovy statements into objects that can be made available to Java DATA Step Objects
- Can include any necessary JAR files with CLASSPATH statement

Accessing the Metadata from Define-XML

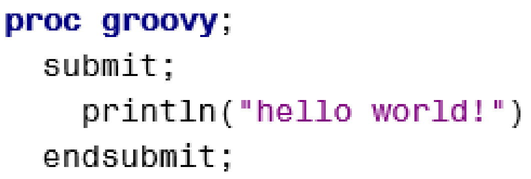
Groovy

- Groovy code submitted with PROC GROOVY runs as process owner
- Process owner has access to resources (file system, network, registry, ...)
- This may be a security issue inside multi-user server
- PROC GROOVY runs only if the **NOXCMD** is turned off
- Not available with the SAS University Edition

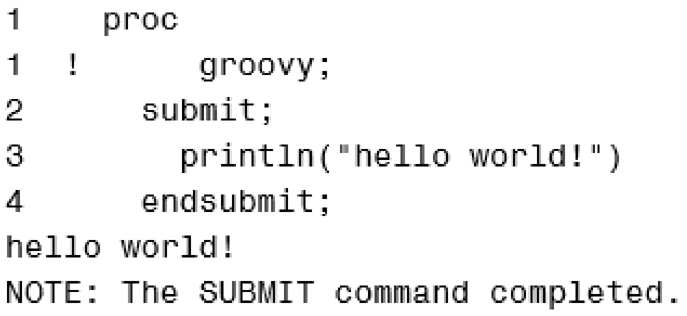
Accessing the Metadata from Define-XML

Groovy syntax

- PROC GROOVY executes Groovy code between SUBMIT and ENDSUBMIT statements.



```
proc groovy;
  submit;
    println("hello world!")
  endsubmit;
```



```
1     proc
1   !      groovy;
2       submit;
3         println("hello world!")
4       endsubmit;
hello world!
NOTE: The SUBMIT command completed.
```

Accessing the Metadata from Define-XML

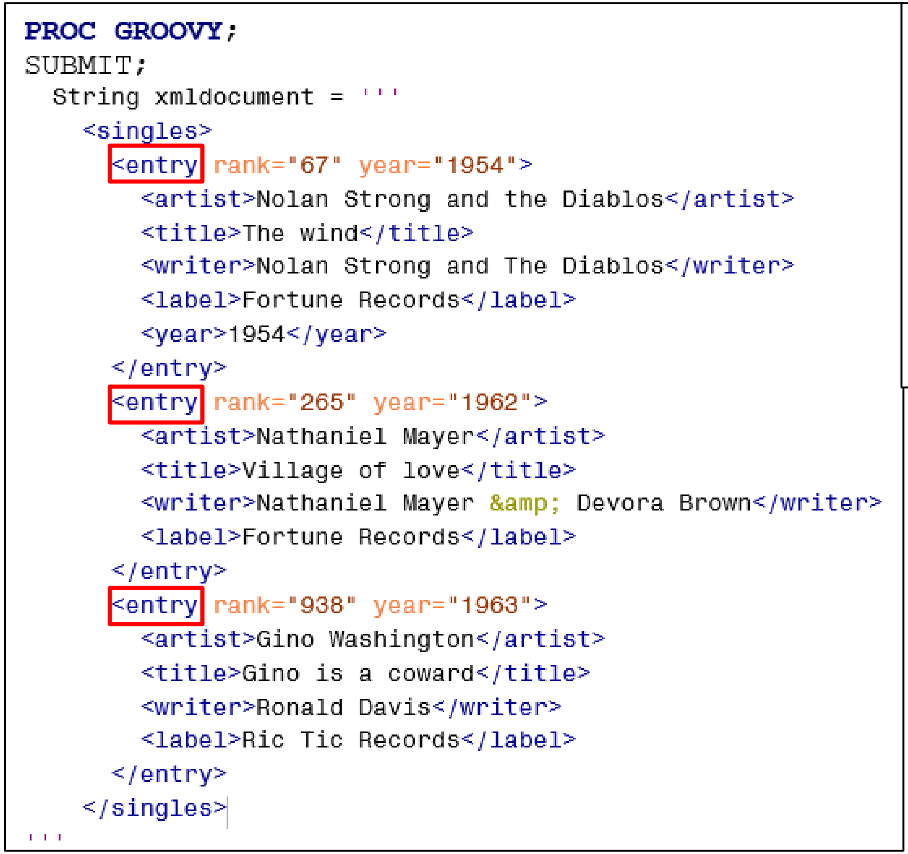
Groovy - interfacing with the DATA Step

- We use PROC GROOVY in order to parse a Define-XML document
- Once parsed we create a CSV file using the CSVWriter class from opencsv (http://opencsv.sourceforge.net/)
  - Ignoring commas in quoted elements.
  - Handling quoted entries with embedded carriage returns (i.e. entries that span multiple lines).
  - Configurable separator and quote characters (or use sensible defaults).
  - Reading and writing from an array of strings
  - Read or write all the entries at once, or use an Iterator-style model

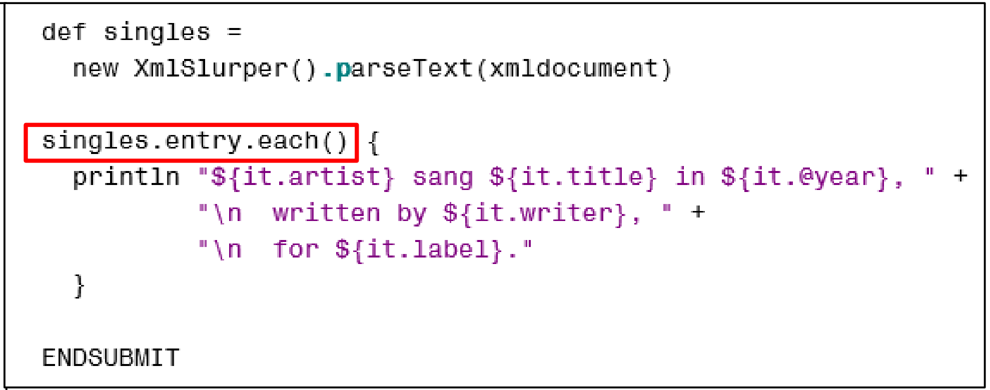
Accessing the Metadata from Define-XML

Groovy and XML

- Groovy provides the **XmlSlurper** class (groovy.util.XmlSlurper) to process XML
- **XmlSlurper** parses an XML document into a **GPathResult** object
- **Gpath** can identify nested structured data (like **XPath** in **XML**)
- <u>Examples</u>:
  - `Study.MetaDataVersion.ItemGroupDef.ItemRef`
  - Attributes: `ItemDef.@DataType`
`ItemDef.@'def:DisplayFormat'`



```
PROC GROOVY;
SUBMIT;
  String xmldocument = '''
    <singles>
      <entry rank="67" year="1954">
        <artist>Nolan Strong and the Diablos</artist>
        <title>The wind</title>
        <writer>Nolan Strong and The Diablos</writer>
        <label>Fortune Records</label>
        <year>1954</year>
      </entry>
      <entry rank="265" year="1962">
        <artist>Nathaniel Mayer</artist>
        <title>Village of love</title>
        <writer>Nathaniel Mayer &amp; Devora Brown</writer>
        <label>Fortune Records</label>
      </entry>
      <entry rank="938" year="1963">
        <artist>Gino Washington</artist>
        <title>Gino is a coward</title>
        <writer>Ronald Davis</writer>
        <label>Ric Tic Records</label>
      </entry>
    </singles>
'''
```



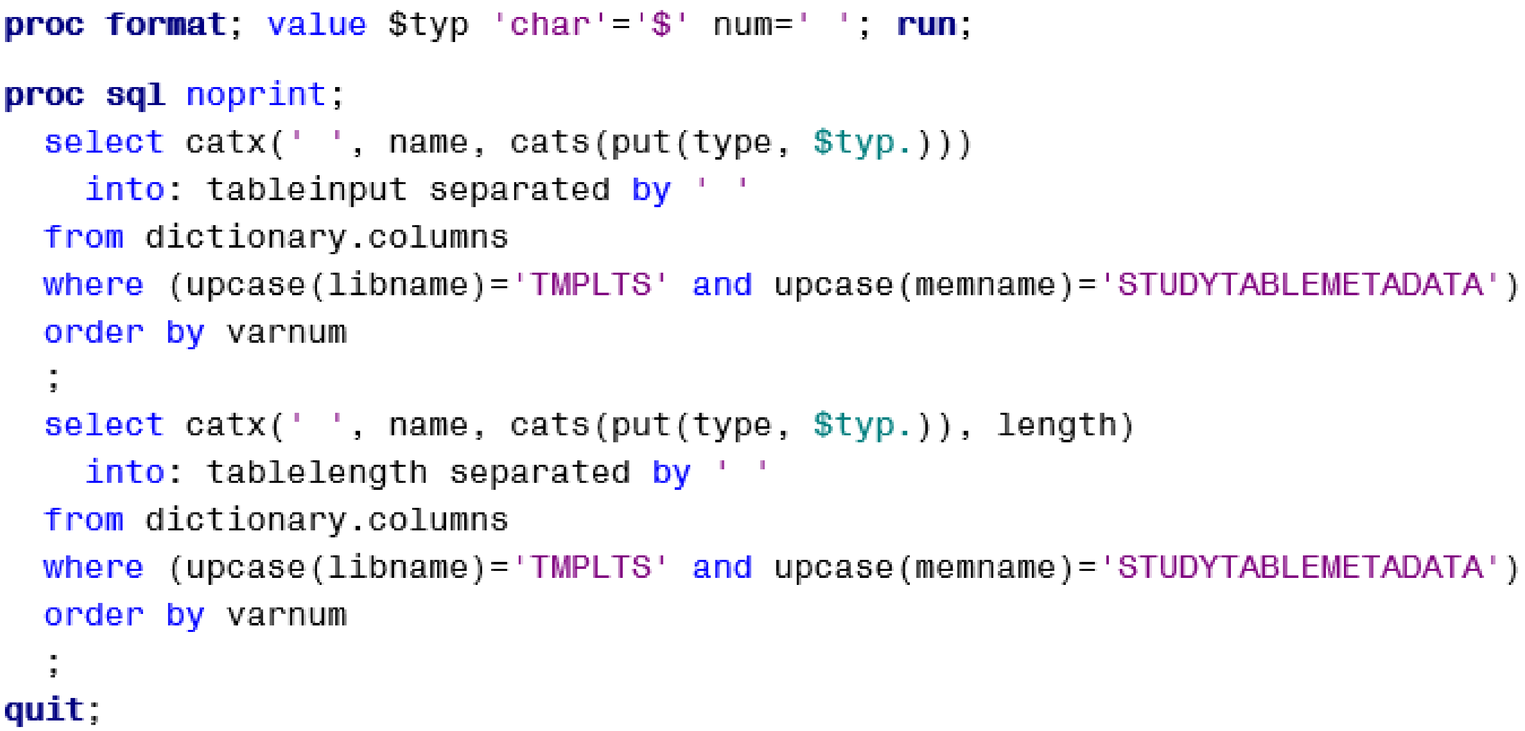
```
def singles =
  new XmlSlurper().parseText(xmldocument)

singles.entry.each() {
  println "${it.artist} sang ${it.title} in ${it.@year}, " +
          "\n  written by ${it.writer}, " +
          "\n  for ${it.label}."
  }

ENDSUBMIT
```

```
Nolan Strong and the Diablos sang The wind in 1954,
  written by Nolan Strong and The Diablos,
  for Fortune Records.
Nathaniel Mayer sang Village of love in 1962,
  written by Nathaniel Mayer & Devora Brown,
  for Fortune Records.
Gino Washington sang Gino is a coward in 1963,
  written by Ronald Davis,
  for Ric Tic Records.
```

Accessing the Metadata from Define-XML

Using the templates

- Use the templates to define data type and lengths for reading CSV

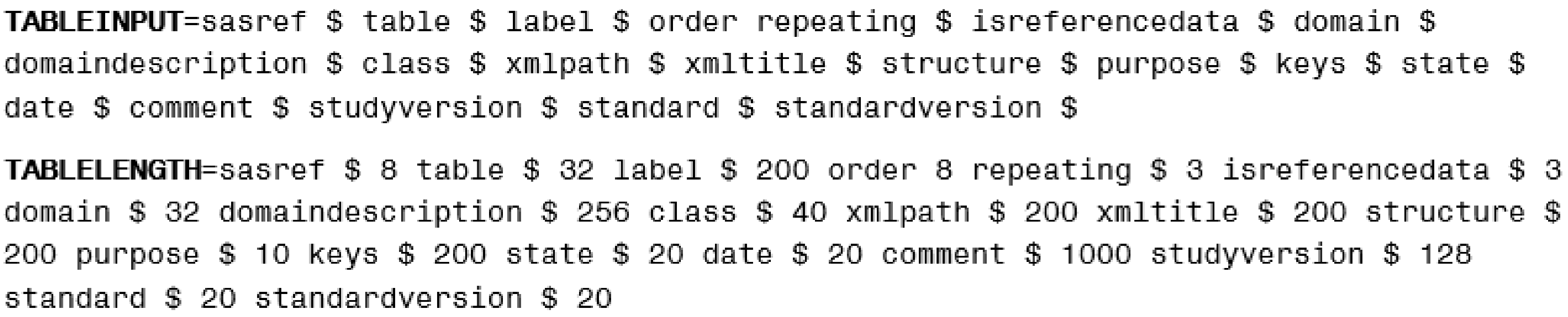


```
proc format; value $typ 'char'='$' num=' '; run;

proc sql noprint;
  select catx(' ', name, cats(put(type, $typ.)))
    into: tableinput separated by ' '
  from dictionary.columns
  where (upcase(libname)='TMPLTS' and upcase(memname)='STUDYTABLEMETADATA')
  order by varnum
  ;
  select catx(' ', name, cats(put(type, $typ.)), length)
    into: tablelength separated by ' '
  from dictionary.columns
  where (upcase(libname)='TMPLTS' and upcase(memname)='STUDYTABLEMETADATA')
  order by varnum
  ;
quit;
```

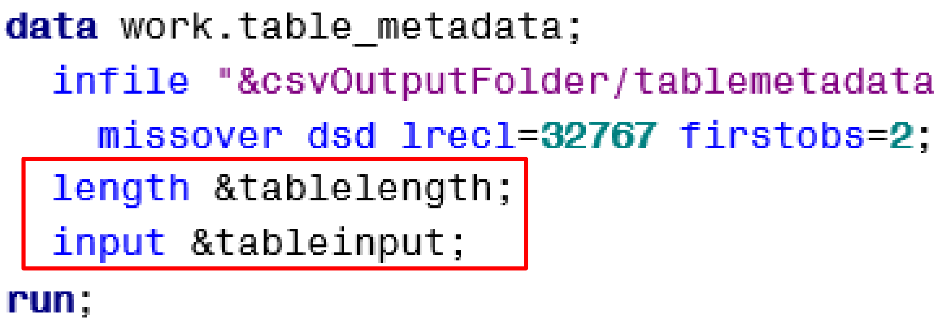
Accessing the Metadata from Define-XML

Using the templates

- This creates 2 macro variables



```
TABLEINPUT=sasref $ table $ label $ order repeating $ isreferencedata $ domain $
domaindescription $ class $ xmlpath $ xmltitle $ structure $ purpose $ keys $ state $
date $ comment $ studyversion $ standard $ standardversion $

TABLELENGTH=sasref $ 8 table $ 32 label $ 200 order 8 repeating $ 3 isreferencedata $ 3
domain $ 32 domaindescription $ 256 class $ 40 xmlpath $ 200 xmltitle $ 200 structure $
200 purpose $ 10 keys $ 200 state $ 20 date $ 20 comment $ 1000 studyversion $ 128
standard $ 20 standardversion $ 20
```



```
data work.table_metadata;
  infile "&csvOutputFolder/tablemetadata.csv" delimiter='09'x
    missover dsd lrecl=32767 firstobs=2;
  length &tablelength;
  input &tableinput;
run;
```

Transforming XML with PROC XSL

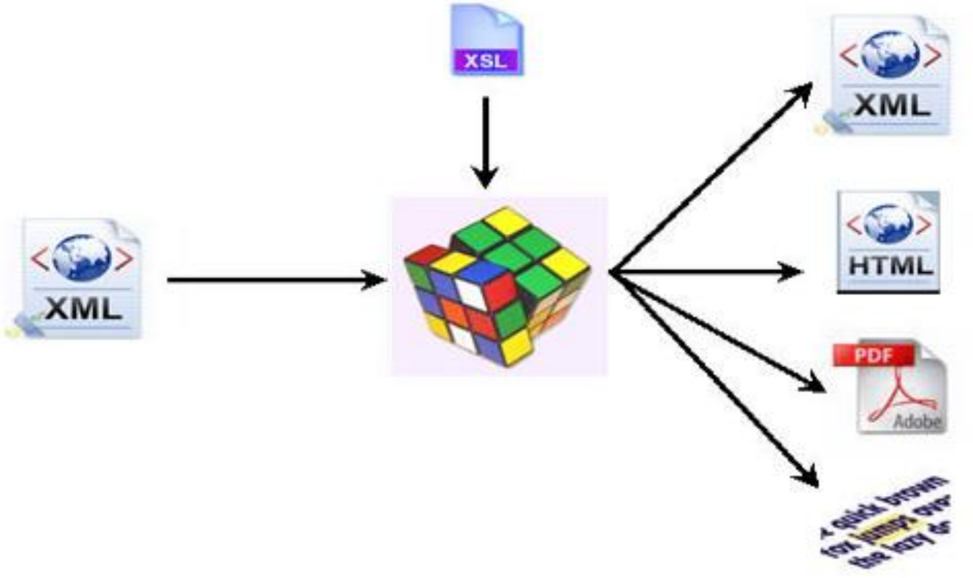
Accessing the Metadata from Define-XML

XSLT

- **XSLT** (eXtensible Stylesheet Language Transformations) is a language that lets you convert **XML** documents into other **XML** documents, into **HTML** documents, or into any other **text** based document, or even a **PDF** file
- XSLT is a language "for transforming the structure and content of an XML document“
- The original XML document **must be well formed**
- XML focuses on **content** and **structure** while XSLT focuses on **presentation**
- To do a transformation you need an XSLT processor
- Most Web browsers have a built-in XSLT processor

Accessing the Metadata from Define-XML

XSLT



- XML
- HTML
- PDF
- TEXT (.SAS, .SQL, ...)

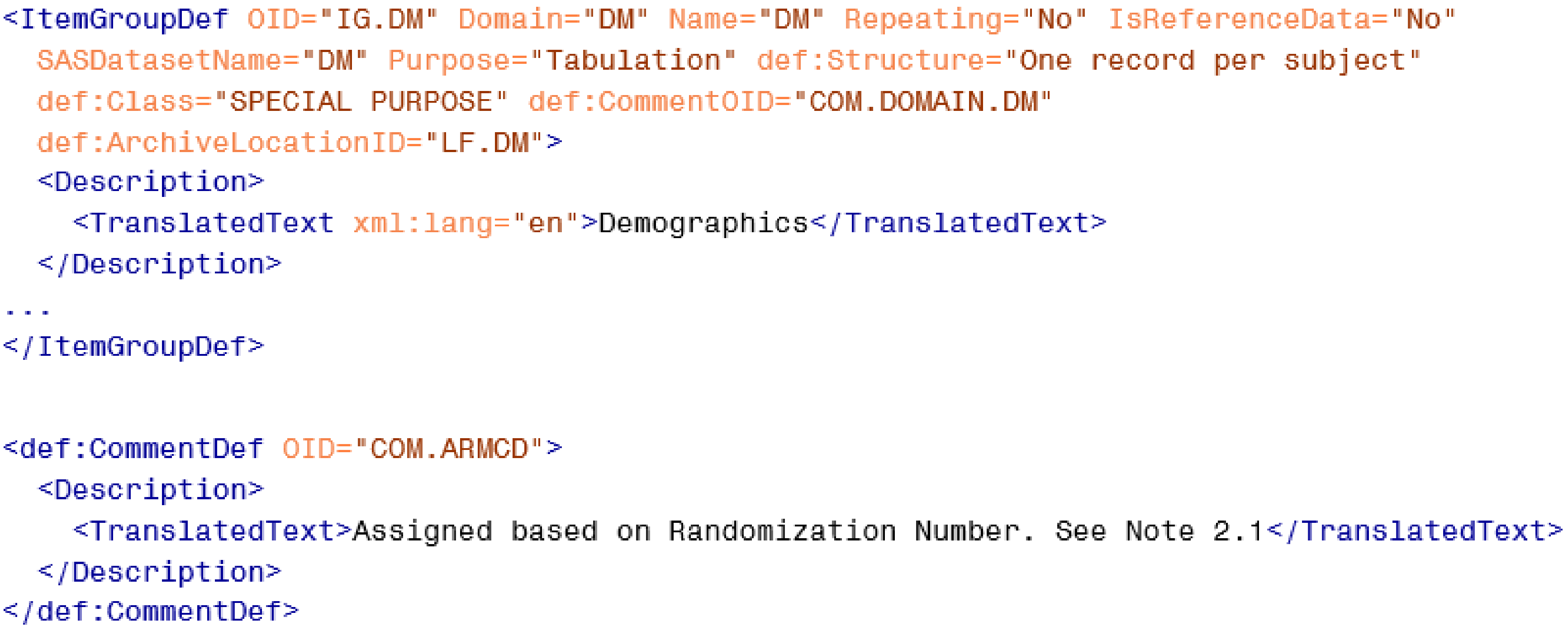
Accessing the Metadata from Define-XML

The import process using XSLT

1. Write an XSL transformation to "flatten" the Define-XML document from a **complex** XML file to a **simple** XML file
2. Use the metadata templates to automatically create an **XMLMap**
3. Use the XML engine with the XMLMap to read the **simple** XML into SAS datasets

Accessing the Metadata from Define-XML

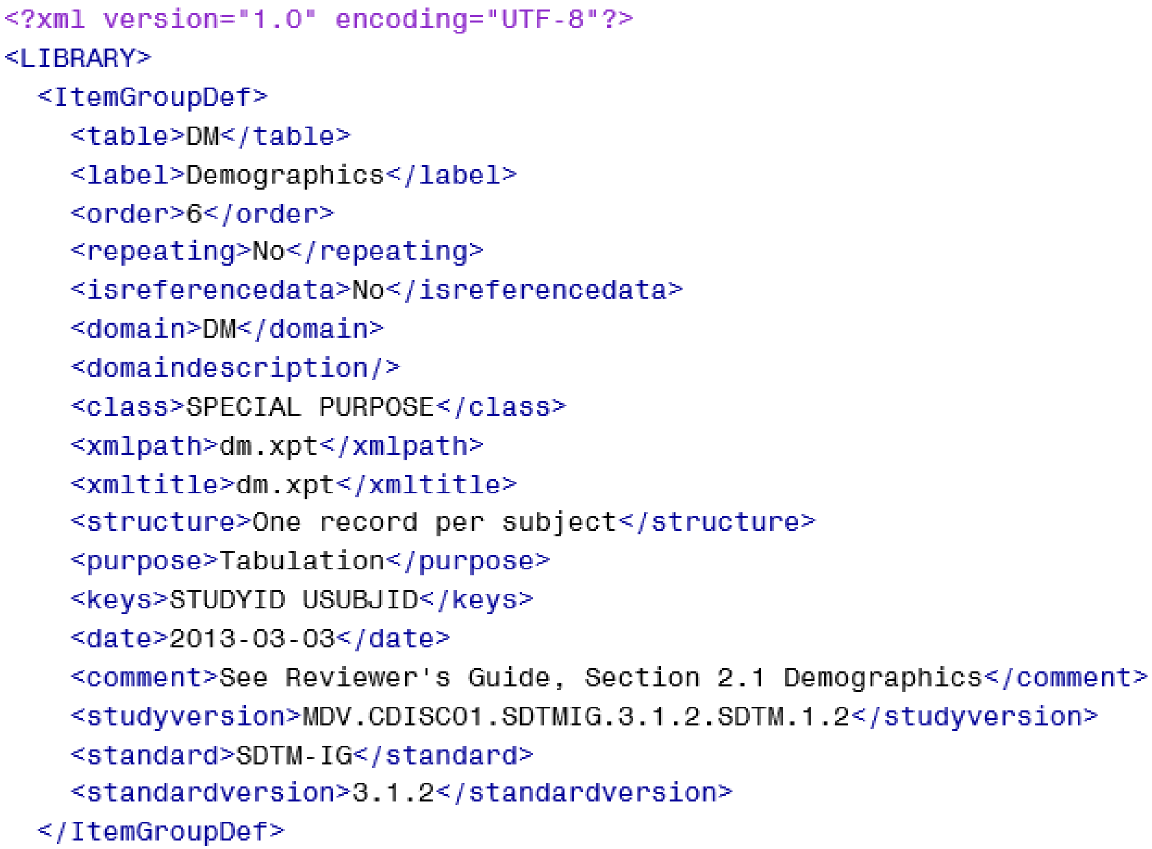
"flatten" the Define-XML document



```
<ItemGroupDef OID="IG.DM" Domain="DM" Name="DM" Repeating="No" IsReferenceData="No"
  SASDatasetName="DM" Purpose="Tabulation" def:Structure="One record per subject"
  def:Class="SPECIAL PURPOSE" def:CommentOID="COM.DOMAIN.DM"
  def:ArchiveLocationID="LF.DM">
  <Description>
    <TranslatedText xml:lang="en">Demographics</TranslatedText>
  </Description>
...
</ItemGroupDef>


<def:CommentDef OID="COM.ARMCD">
  <Description>
    <TranslatedText>Assigned based on Randomization Number. See Note 2.1</TranslatedText>
  </Description>
</def:CommentDef>
```

Accessing the Metadata from Define-XML

"flatten" the Define-XML document

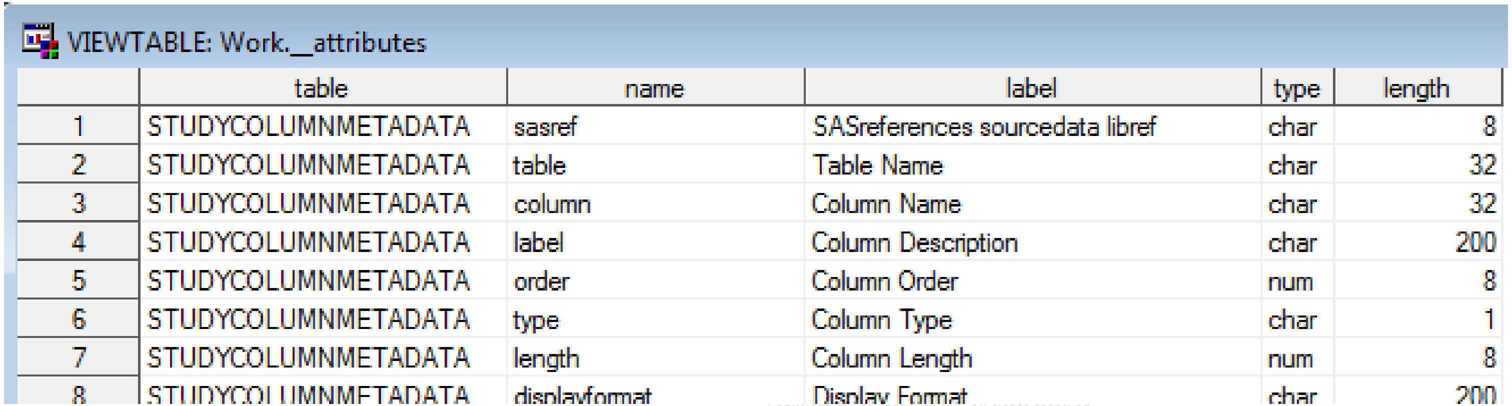


```
<?xml version="1.0" encoding="UTF-8"?>
<LIBRARY>
  <ItemGroupDef>
    <table>DM</table>
    <label>Demographics</label>
    <order>6</order>
    <repeating>No</repeating>
    <isreferencedata>No</isreferencedata>
    <domain>DM</domain>
    <domaindescription/>
    <class>SPECIAL PURPOSE</class>
    <xmlpath>dm.xpt</xmlpath>
    <xmltitle>dm.xpt</xmltitle>
    <structure>One record per subject</structure>
    <purpose>Tabulation</purpose>
    <keys>STUDYID USUBJID</keys>
    <date>2013-03-03</date>
    <comment>See Reviewer's Guide, Section 2.1 Demographics</comment>
    <studyversion>MDV.CDISC01.SDTMIG.3.1.2.SDTM.1.2</studyversion>
    <standard>SDTM-IG</standard>
    <standardversion>3.1.2</standardversion>
  </ItemGroupDef>
```

Accessing the Metadata from Define-XML

Read the "flat" XML with an XMLMap

```
proc sql noprint;
  create table work.__attributes as
  select memname as table, strip(name) as name, strip(label) as label, type, length
  from dictionary.columns
  where libname="TMPLTS" and memname="STUDYCOLUMNMETADATA"
  order by varnum
  ;
quit;
```

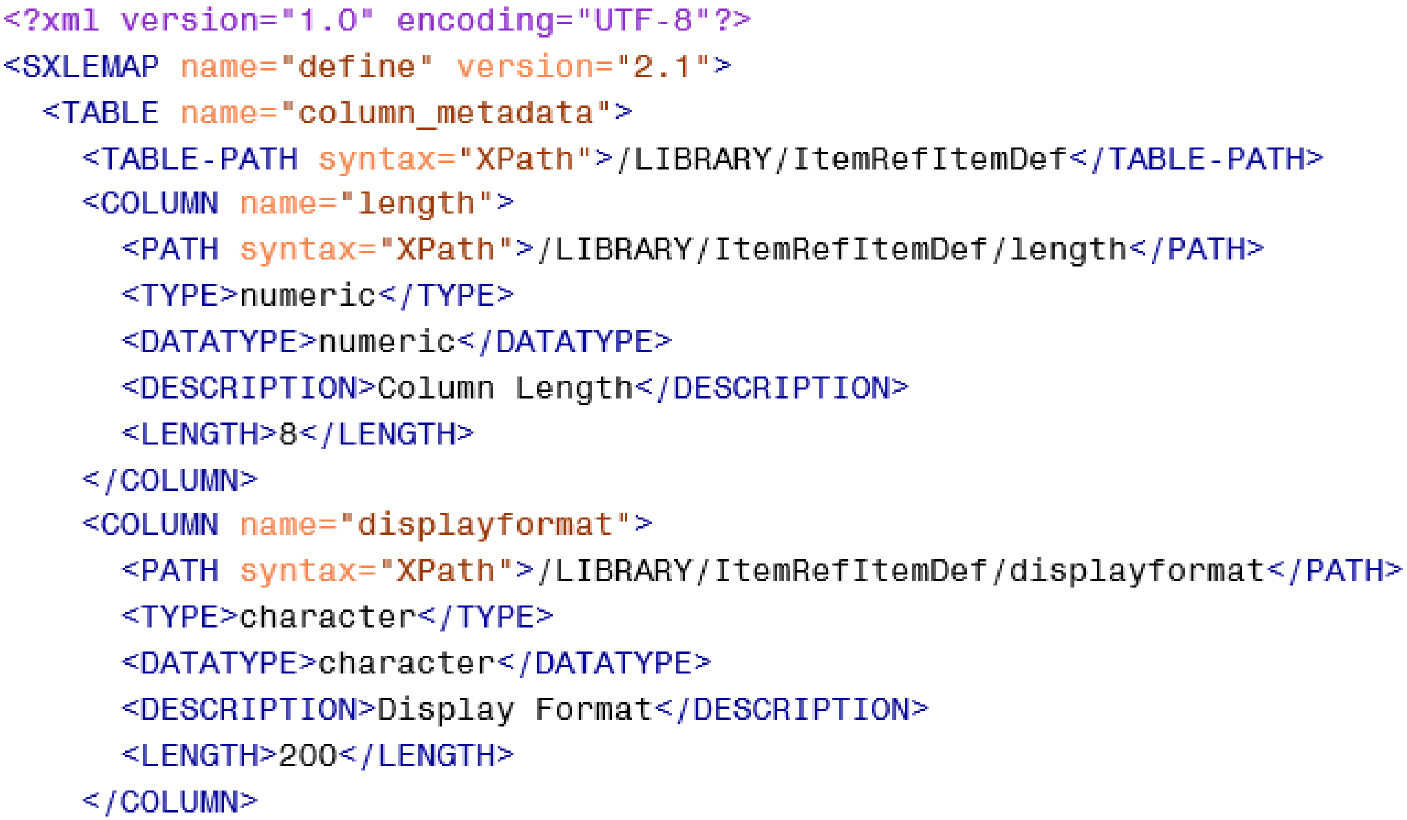


VIEWTABLE: Work.__attributes

	table	name	label	type	length
1	STUDYCOLUMNMETADATA	sasref	SASreferences sourcedata libref	char	8
2	STUDYCOLUMNMETADATA	table	Table Name	char	32
3	STUDYCOLUMNMETADATA	column	Column Name	char	32
4	STUDYCOLUMNMETADATA	label	Column Description	char	200
5	STUDYCOLUMNMETADATA	order	Column Order	num	8
6	STUDYCOLUMNMETADATA	type	Column Type	char	1
7	STUDYCOLUMNMETADATA	length	Column Length	num	8
8	STUDYCOLUMNMETADATA	displayformat	Display Format	char	200

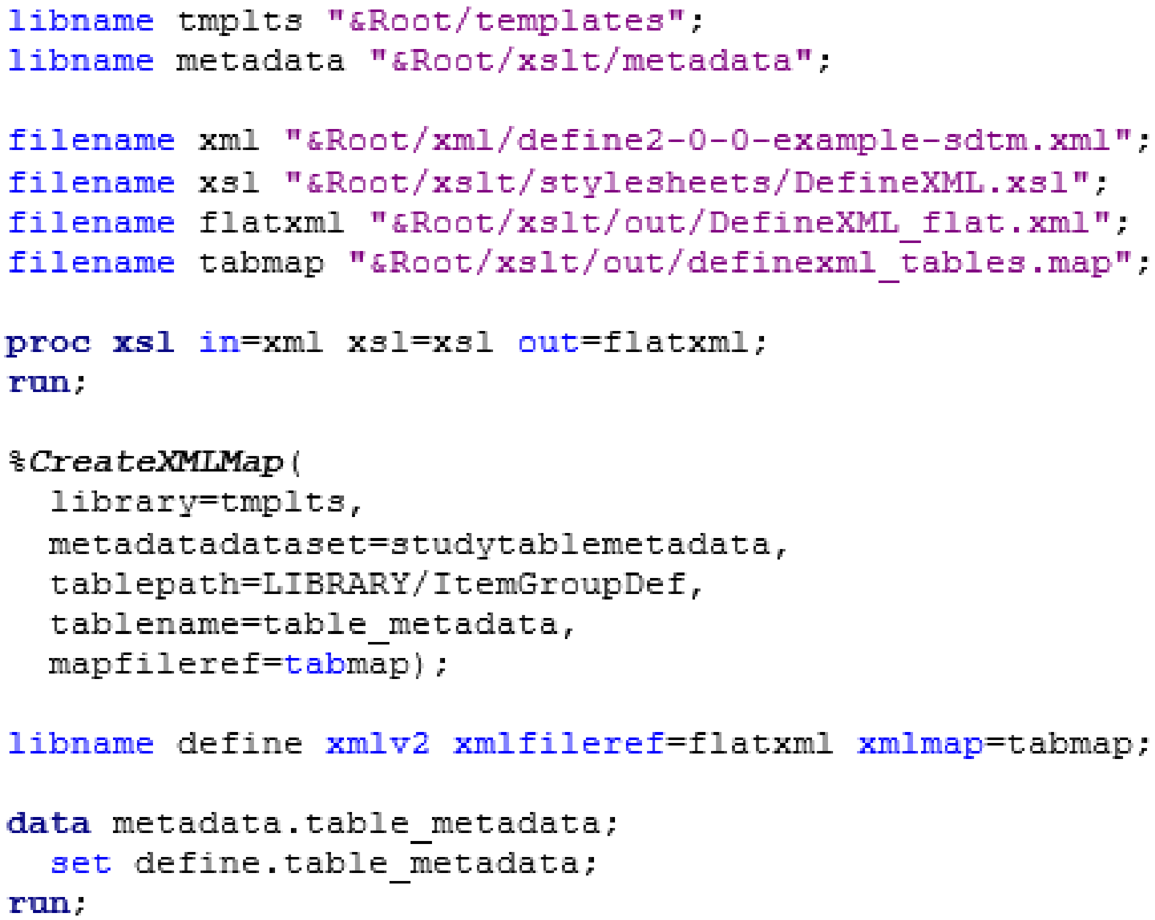
Accessing the Metadata from Define-XML

Read the "flat" XML with an XMLMap



```xml
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="define" version="2.1">
  <TABLE name="column_metadata">
    <TABLE-PATH syntax="XPath">/LIBRARY/ItemRefItemDef</TABLE-PATH>
    <COLUMN name="length">
      <PATH syntax="XPath">/LIBRARY/ItemRefItemDef/length</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>numeric</DATATYPE>
      <DESCRIPTION>Column Length</DESCRIPTION>
      <LENGTH>8</LENGTH>
    </COLUMN>
    <COLUMN name="displayformat">
      <PATH syntax="XPath">/LIBRARY/ItemRefItemDef/displayformat</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>character</DATATYPE>
      <DESCRIPTION>Display Format</DESCRIPTION>
      <LENGTH>200</LENGTH>
    </COLUMN>
```

Accessing the Metadata from Define-XML



```
libname tmplts "&Root/templates";
libname metadata "&Root/xslt/metadata";

filename xml "&Root/xml/define2-0-0-example-sdtm.xml";
filename xsl "&Root/xslt/stylesheets/DefineXML.xsl";
filename flatxml "&Root/xslt/out/DefineXML_flat.xml";
filename tabmap "&Root/xslt/out/definexml_tables.map";

proc xsl in=xml xsl=xsl out=flatxml;
run;

%CreateXMLMap(
  library=tmplts,
  metadatadataset=studytablemetadata,
  tablepath=LIBRARY/ItemGroupDef,
  tablename=table_metadata,
  mapfileref=tabmap);

libname define xmlv2 xmlfileref=flatxml xmlmap=tabmap;

data metadata.table_metadata;
  set define.table_metadata;
run;
```

SAS Clinical Standards Toolkit

SAS Clinical Standards Toolkit 1.7

Introduction

- Framework to primarily support Clinical Research activities (CDISC).
- **Designed** to customize and extend
- A collection of metadata and "tools", providing an initial set of standards and functionality that is evolving and growing with updates and releases.
- Provides **SAS representation of published standards** as SAS data sets and catalogs
  - **Content** standards: SDTM, ADaM, SEND
  - **XML** standards: Define-XML, Dataset-XML and ODM
  - **Controlled Terminology** standards (CDISC/NCI)

SAS Clinical Standards Toolkit 1.7

Introduction

- Supported CDISC standards in SAS Clinical Standards Toolkit 1.7:
  - SDTM 3.1.2, 3.1.3 and 3.2
  - ADaM 2.1 (ADSL, Basic Data Structure, ADAE and ADTTE)
  - CDASH 1.1 Domain definitions
  - SEND 3.0 (initial implementation)
  - CRT-DDS 1.0 (Define-XML - Create / Import / Validate)
  - Define-XML 2.0 (Create / Import / Validate), including Analysis Results Metadata
  - Dataset-XML 1.0 (Create / Import / Validate)
  - ODM 1.3.0, 1.3.1 - Read / Write / Validate
  - NCI CDISC Controlled Terminology (June 2014)
    (import/export of ODM XML through CT 1.0 standard)

SAS Clinical Standards Toolkit 1.7

Introduction

- Hotfix for Toolkit 1.7 (**CST 1.7.1**):
  - Support for Analysis Results Metadata v1.0 for Define-XML v2
  - CST 1.7.1 available since October 2016.
- Latest Hotfix for Toolkit 1.7 (**CST 1.7.2**):
  - Bug fixes
  - Customizable metadata support for:
    - ODM/@FileOID, Study/@OID,
    - ItemGroupDef: Repeating and IsReferenceData attributes,
    - MethodDef/@Type, MethodDef/FormalExpression
  - CST 1.7.2 available since December 2017

SAS Clinical Standards Toolkit 1.7

Introduction

- Available to all licensed SAS customers at no additional charge
- Supported with SAS 9.4M3 or later on the following operating systems:
  - Windows x64
  - Linux x64
- Separately orderable component
- Contact your SAS Account Representative concerning availability

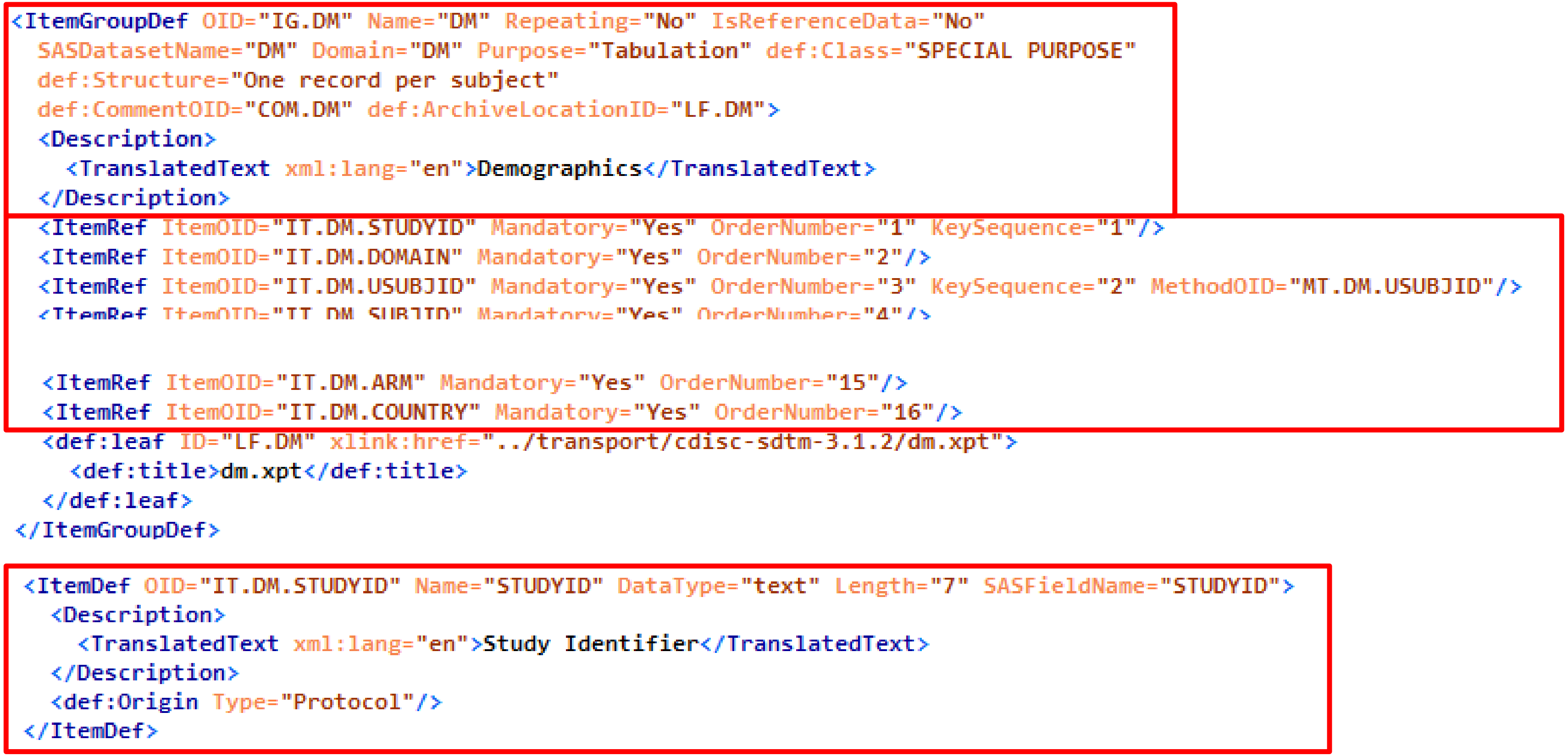
SAS Clinical Standards Toolkit 1.7

SAS Data Model for Define-XML

- SAS Clinical Standards Toolkit v1.7 provides a data model that represents the Define-XML v2 format in SAS data sets
- Patterned to match the XML element and attribute structure of the Define-XML format
  - XML element → table
  - XML attribute → column

SAS Clinical Standards Toolkit 1.7

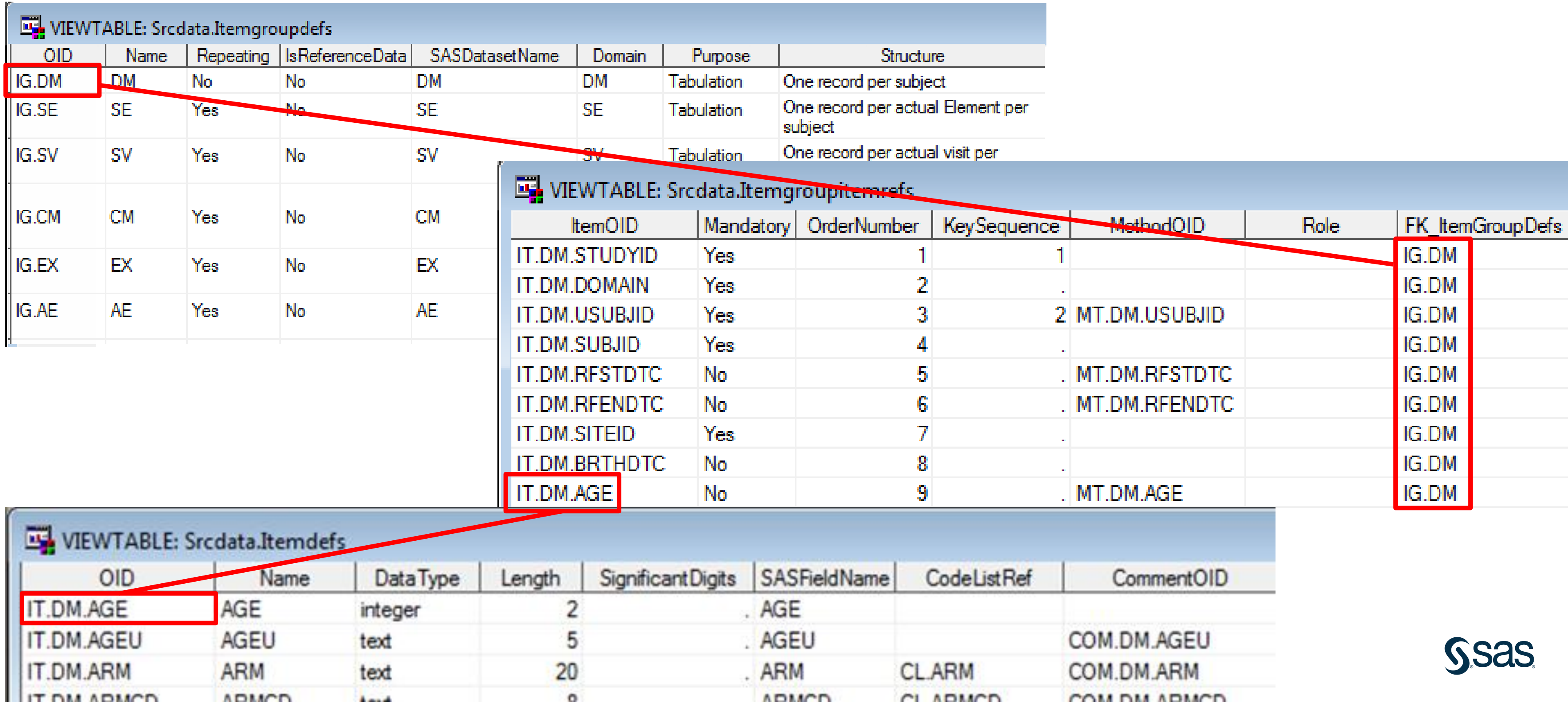
SAS Data Model for Define-XML



```
<ItemGroupDef OID="IG.DM" Name="DM" Repeating="No" IsReferenceData="No"
  SASDatasetName="DM" Domain="DM" Purpose="Tabulation" def:Class="SPECIAL PURPOSE"
  def:Structure="One record per subject"
  def:CommentOID="COM.DM" def:ArchiveLocationID="LF.DM">
  <Description>
    <TranslatedText xml:lang="en">Demographics</TranslatedText>
  </Description>
  <ItemRef ItemOID="IT.DM.STUDYID" Mandatory="Yes" OrderNumber="1" KeySequence="1"/>
  <ItemRef ItemOID="IT.DM.DOMAIN" Mandatory="Yes" OrderNumber="2"/>
  <ItemRef ItemOID="IT.DM.USUBJID" Mandatory="Yes" OrderNumber="3" KeySequence="2" MethodOID="MT.DM.USUBJID"/>
  <ItemRef ItemOID="IT.DM.SUBJID" Mandatory="Yes" OrderNumber="4"/>


   <ItemRef ItemOID="IT.DM.ARM" Mandatory="Yes" OrderNumber="15"/>
   <ItemRef ItemOID="IT.DM.COUNTRY" Mandatory="Yes" OrderNumber="16"/>
   <def:leaf ID="LF.DM" xlink:href="../transport/cdisc-sdtm-3.1.2/dm.xpt">
     <def:title>dm.xpt</def:title>
   </def:leaf>
 </ItemGroupDef>

 <ItemDef OID="IT.DM.STUDYID" Name="STUDYID" DataType="text" Length="7" SASFieldName="STUDYID">
   <Description>
     <TranslatedText xml:lang="en">Study Identifier</TranslatedText>
   </Description>
   <def:Origin Type="Protocol"/>
 </ItemDef>
```

SAS Clinical Standards Toolkit 1.7

SAS Data Model for Define-XML



VIEWTABLE: Srcdata.Itemgroupdefs

OID	Name	Repeating	IsReferenceData	SASDatasetName	Domain	Purpose	Structure
IG.DM	DM	No	No	DM	DM	Tabulation	One record per subject
IG.SE	SE	Yes	No	SE	SE	Tabulation	One record per actual Element per subject
IG.SV	SV	Yes	No	SV	SV	Tabulation	One record per actual visit per
IG.CM	CM	Yes	No	CM			
IG.EX	EX	Yes	No	EX			
IG.AE	AE	Yes	No	AE			

VIEWTABLE: Srcdata.Itemgroupitemrefs

ItemOID	Mandatory	OrderNumber	KeySequence	MethodOID	Role	FK_ItemGroupDefs
IT.DM.STUDYID	Yes	1	1			IG.DM
IT.DM.DOMAIN	Yes	2	.			IG.DM
IT.DM.USUBJID	Yes	3	2	MT.DM.USUBJID		IG.DM
IT.DM.SUBJID	Yes	4	.			IG.DM
IT.DM.RFSTDTC	No	5	.	MT.DM.RFSTDTC		IG.DM
IT.DM.RFENDTC	No	6	.	MT.DM.RFENDTC		IG.DM
IT.DM.SITEID	Yes	7	.			IG.DM
IT.DM.BRTHDTC	No	8	.			IG.DM
IT.DM.AGE	No	9	.	MT.DM.AGE		IG.DM

VIEWTABLE: Srcdata.Itemdefs

OID	Name	DataType	Length	SignificantDigits	SASFieldName	CodeListRef	CommentOID
IT.DM.AGE	AGE	integer	2	.	AGE		
IT.DM.AGEU	AGEU	text	5	.	AGEU		COM.DM.AGEU
IT.DM.ARM	ARM	text	20	.	ARM	CL.ARM	COM.DM.ARM
IT.DM.ARMCD	ARMCD	text	8	.	ARMCD	CL.ARMCD	COM.DM.ARMCD

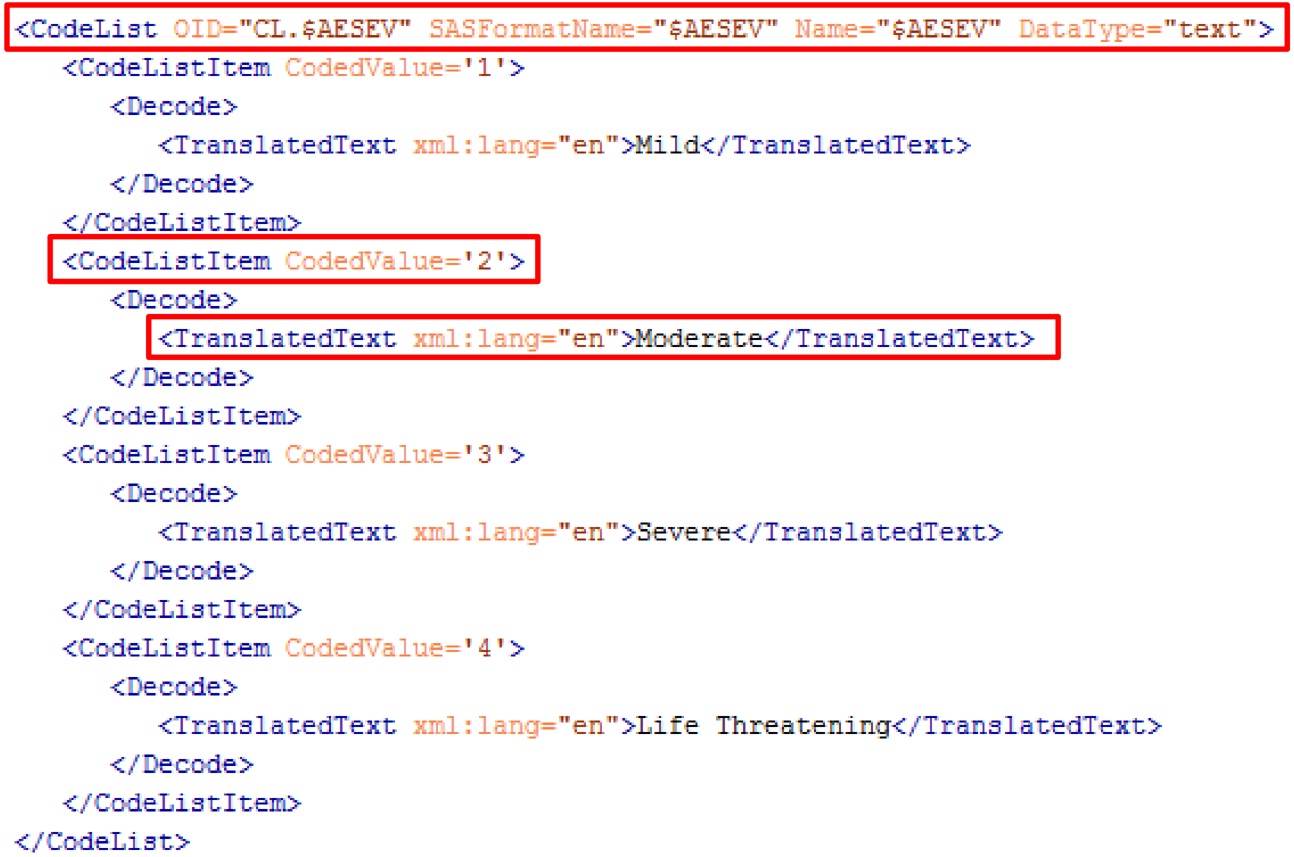
SAS Clinical Standards Toolkit 1.7

SAS Data Model for Define-XML

- Reading and writing Define-XML uses an intermediate 'flat' XML Cube
- This 'flat' XML Cube can be easily transformed to the 2-dimensional SAS data sets

SAS Clinical Standards Toolkit 1.7

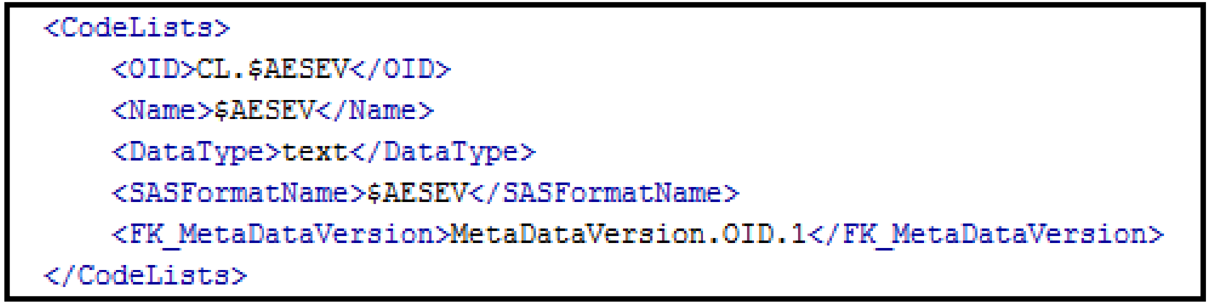
SAS Data Model for Define-XML



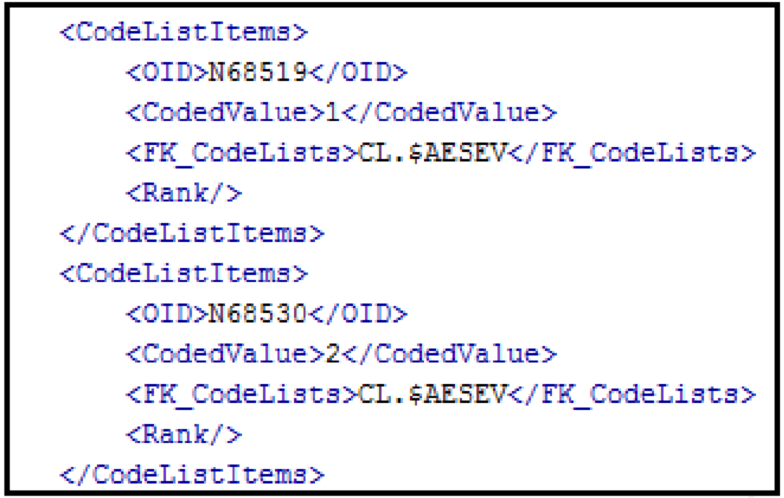
```
<CodeList OID="CL.$AESEV" SASFormatName="$AESEV" Name="$AESEV" DataType="text">
   <CodeListItem CodedValue='1'>
      <Decode>
         <TranslatedText xml:lang="en">Mild</TranslatedText>
      </Decode>
   </CodeListItem>
   <CodeListItem CodedValue='2'>
      <Decode>
         <TranslatedText xml:lang="en">Moderate</TranslatedText>
      </Decode>
   </CodeListItem>
   <CodeListItem CodedValue='3'>
      <Decode>
         <TranslatedText xml:lang="en">Severe</TranslatedText>
      </Decode>
   </CodeListItem>
   <CodeListItem CodedValue='4'>
      <Decode>
         <TranslatedText xml:lang="en">Life Threatening</TranslatedText>
      </Decode>
   </CodeListItem>
</CodeList>
```

SAS Clinical Standards Toolkit 1.7

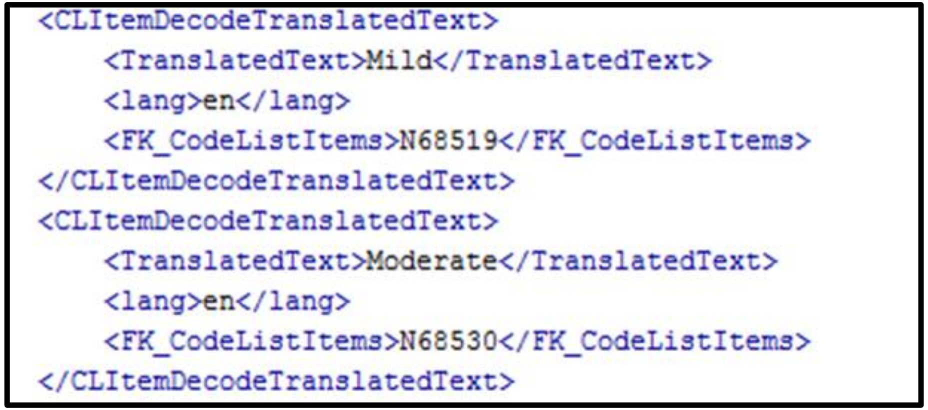
SAS Data Model for Define-XML



```
<CodeLists>
    <OID>CL.$AESEV</OID>
    <Name>$AESEV</Name>
    <DataType>text</DataType>
    <SASFormatName>$AESEV</SASFormatName>
    <FK_MetaDataVersion>MetaDataVersion.OID.1</FK_MetaDataVersion>
</CodeLists>
```



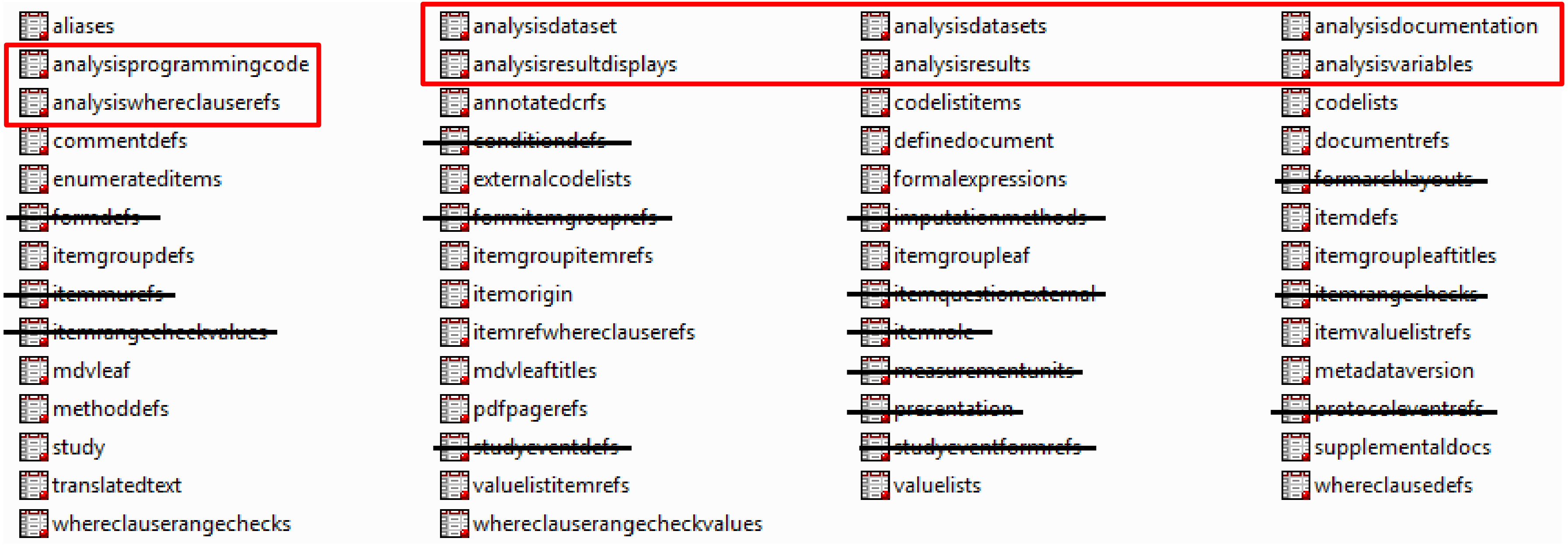
```
<CodeListItems>
    <OID>N68519</OID>
    <CodedValue>1</CodedValue>
    <FK_CodeLists>CL.$AESEV</FK_CodeLists>
    <Rank/>
</CodeListItems>
<CodeListItems>
    <OID>N68530</OID>
    <CodedValue>2</CodedValue>
    <FK_CodeLists>CL.$AESEV</FK_CodeLists>
    <Rank/>
</CodeListItems>
```



```
<CLItemDecodeTranslatedText>
    <TranslatedText>Mild</TranslatedText>
    <lang>en</lang>
    <FK_CodeListItems>N68519</FK_CodeListItems>
</CLItemDecodeTranslatedText>
<CLItemDecodeTranslatedText>
    <TranslatedText>Moderate</TranslatedText>
    <lang>en</lang>
    <FK_CodeListItems>N68530</FK_CodeListItems>
</CLItemDecodeTranslatedText>
```

SAS Clinical Standards Toolkit 1.7

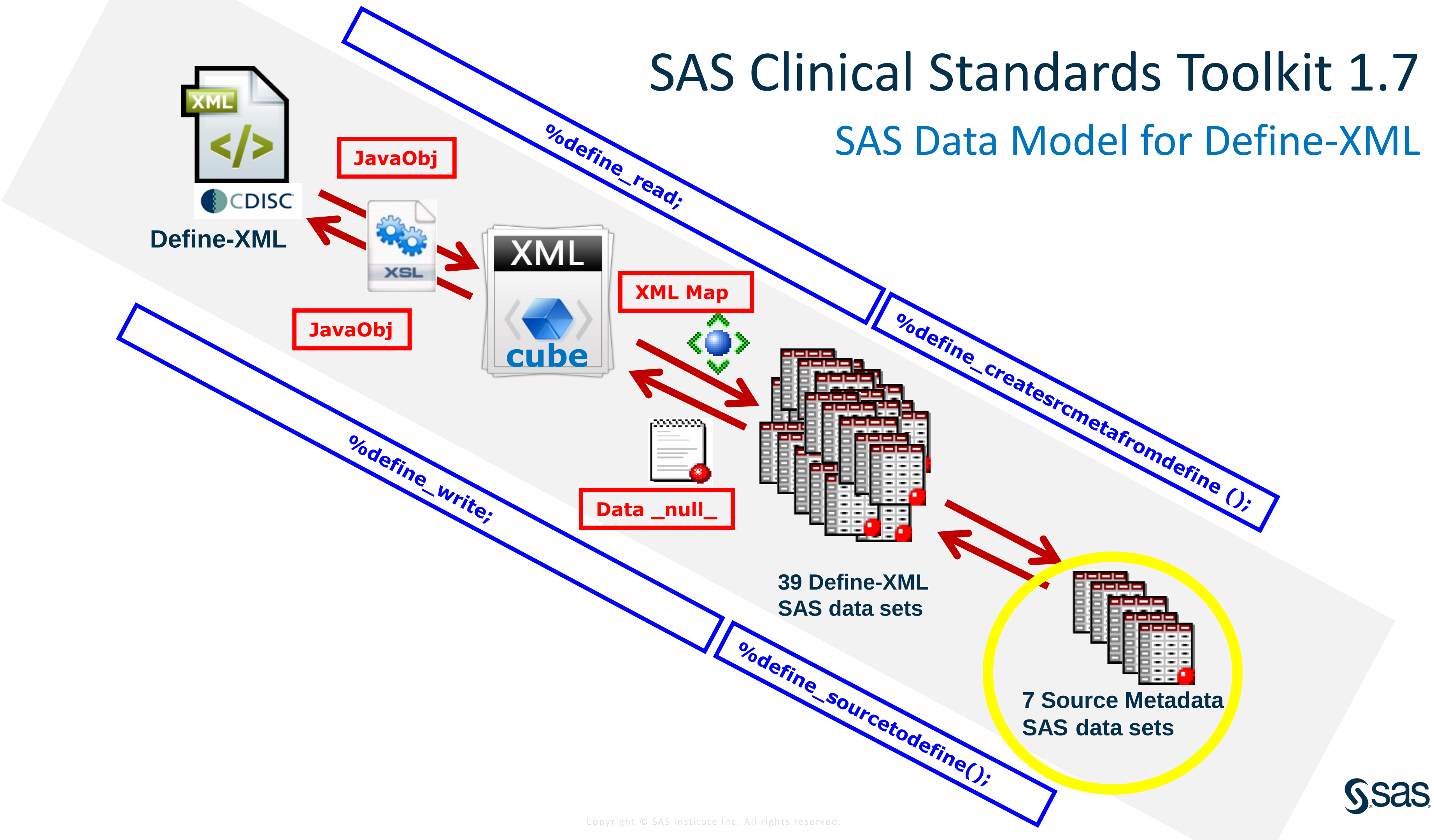
SAS Data Model for Define-XML



- aliases
- analysisdataset
- analysisdatasets
- analysisdocumentation
- analysisprogrammingcode
- analysisresultdisplays
- analysisresults
- analysisvariables
- analysiswhereclauserefs
- annotatedcrfs
- codelistitems
- codelists
- commentdefs
- ~~conditiondefs~~
- definedocument
- documentrefs
- enumerateditems
- externalcodelists
- formalexpressions
- ~~formarchlayouts~~
- ~~formdefs~~
- ~~formitemgrouprefs~~
- ~~imputationmethods~~
- itemdefs
- itemgroupdefs
- itemgroupitemrefs
- itemgroupleaf
- itemgroupleaftitles
- ~~itemmurefs~~
- itemorigin
- ~~itemquestionexternal~~
- ~~itemrangechecks~~
- ~~itemrangecheckvalues~~
- itemrefwhereclauserefs
- ~~itemrole~~
- itemvaluelistrefs
- mdvleaf
- mdvleaftitles
- ~~measurementunits~~
- metadataversion
- methoddefs
- pdfpagerefs
- ~~presentation~~
- ~~protocoleventrefs~~
- study
- ~~studyeventdefs~~
- ~~studyeventformrefs~~
- supplementaldocs
- translatedtext
- valuelistitemrefs
- valuelists
- whereclausedefs
- whereclauserangechecks
- whereclauserangecheckvalues

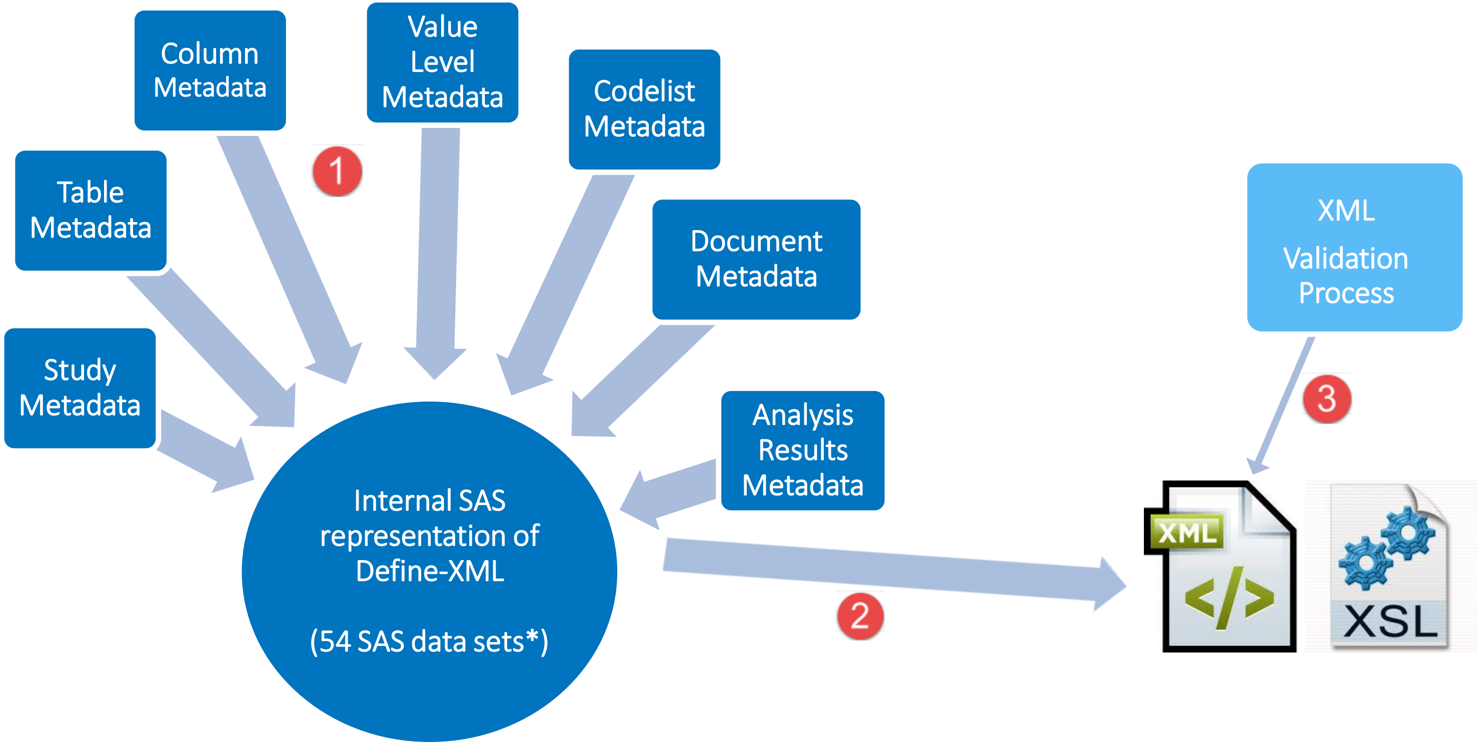
SAS Clinical Standards Toolkit 1.7

SAS Data Model for Define-XML



SAS Clinical Standards Toolkit 1.7

From Study Source Metadata to Define-XML v2



Column Metadata

Value Level Metadata

Codelist Metadata

Table Metadata

Document Metadata

Study Metadata

Analysis Results Metadata

1

Internal SAS representation of Define-XML

(54 SAS data sets*)

2

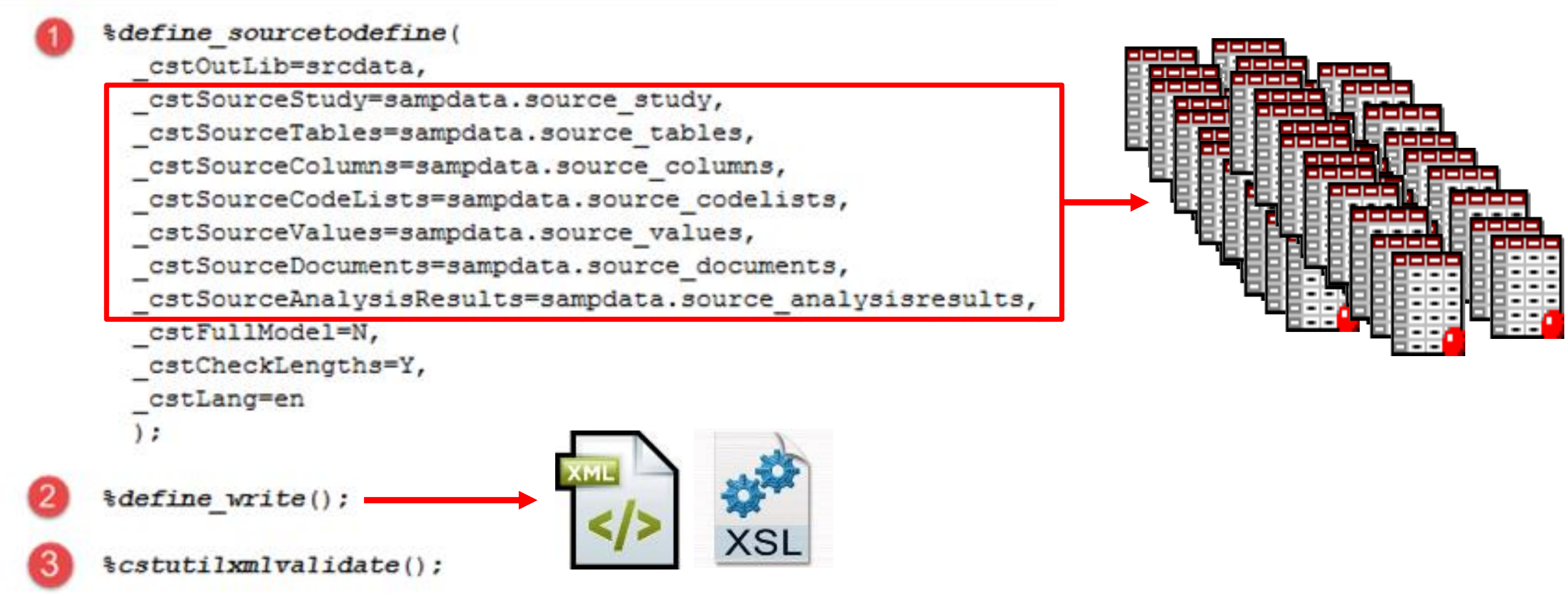
XML Validation Process

3

*Define-XML v2 uses 39 data sets

SAS Clinical Standards Toolkit 1.7.1

From Study Source Metadata to Define-XML v2



1

```
%define_sourcetodefine(
  _cstOutLib=srcdata,
  _cstSourceStudy=sampdata.source_study,
  _cstSourceTables=sampdata.source_tables,
  _cstSourceColumns=sampdata.source_columns,
  _cstSourceCodeLists=sampdata.source_codelists,
  _cstSourceValues=sampdata.source_values,
  _cstSourceDocuments=sampdata.source_documents,
  _cstSourceAnalysisResults=sampdata.source_analysisresults,
  _cstFullModel=N,
  _cstCheckLengths=Y,
  _cstLang=en
  );
```

2

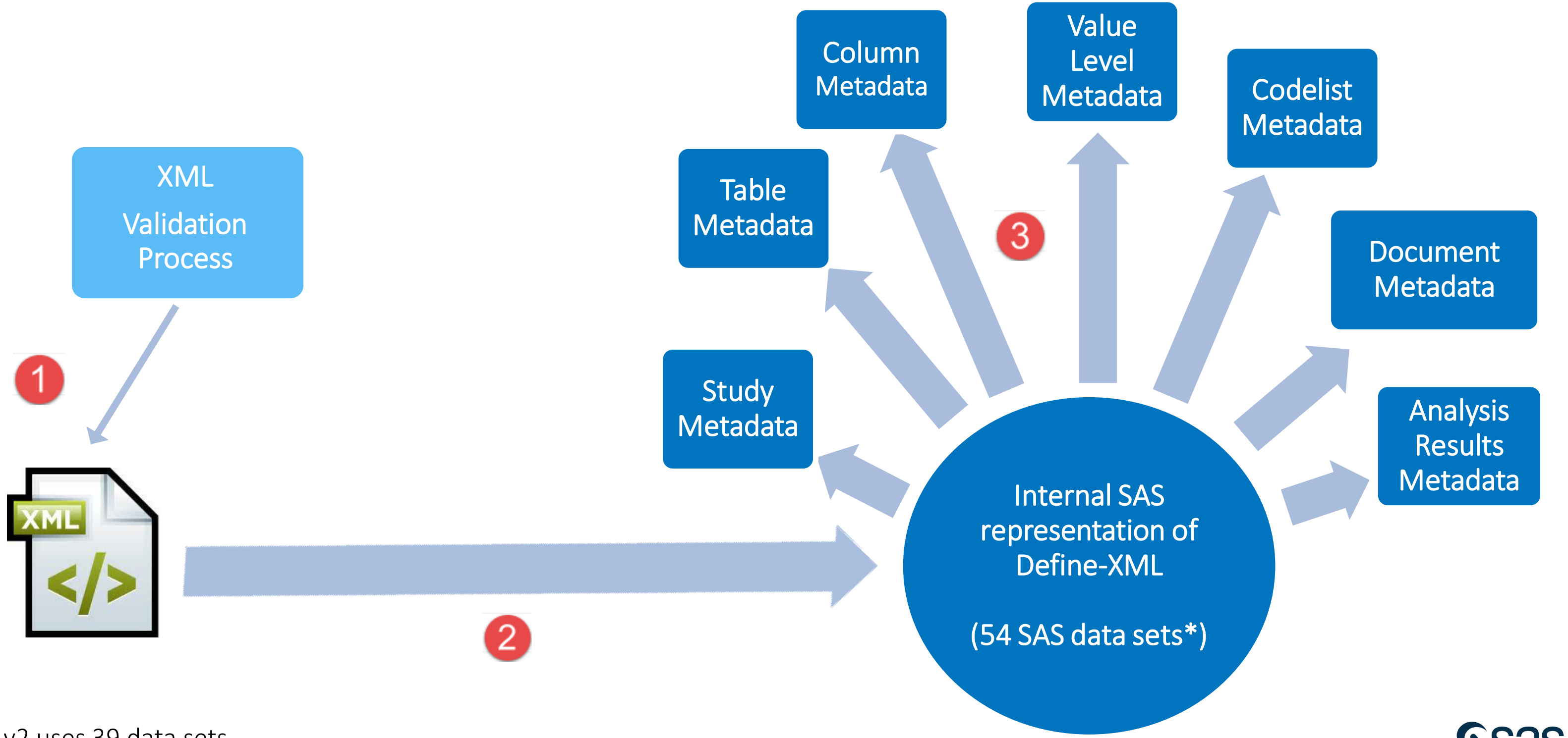
```
%define_write();
```

3

```
%cstutilxmlvalidate();
```

SAS Clinical Standards Toolkit 1.7

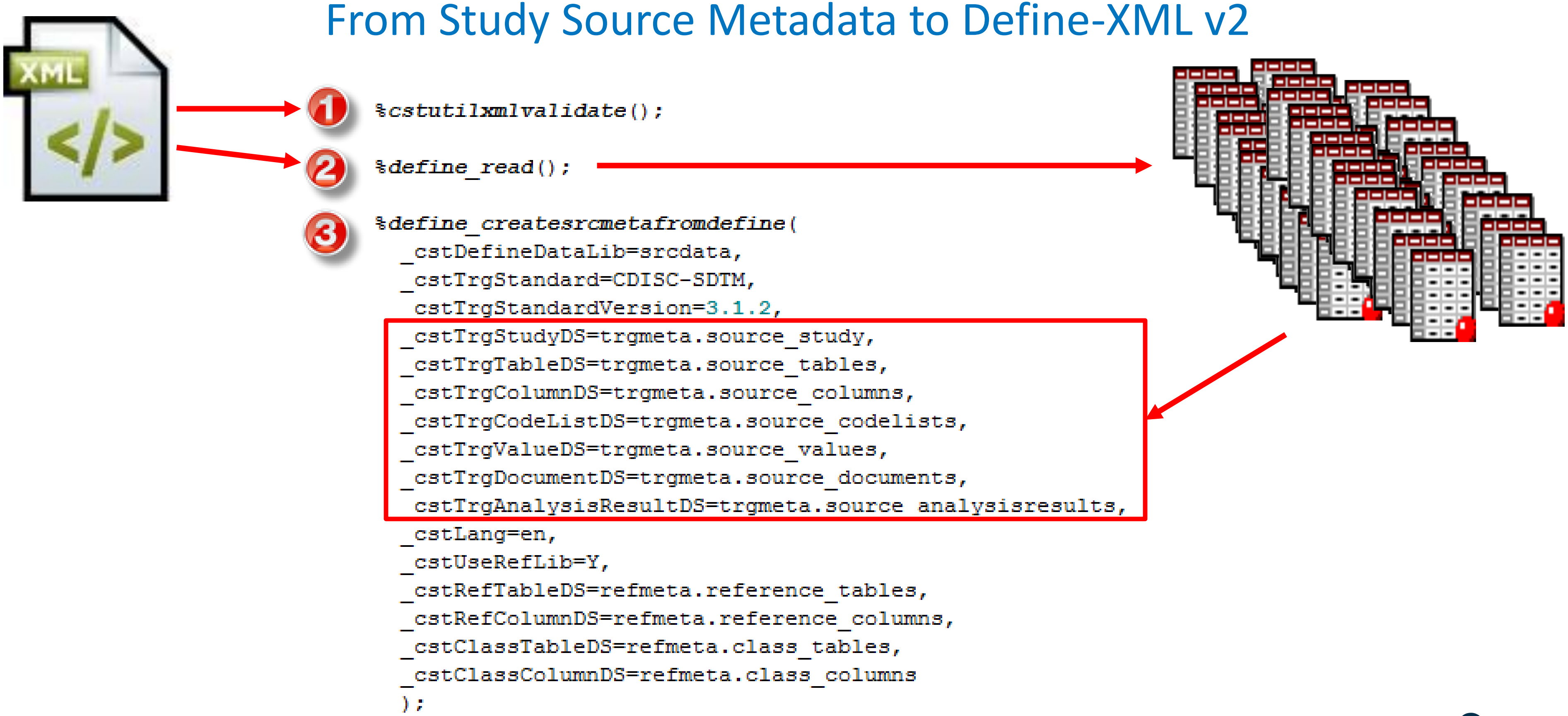
From Define-XML v2 to Study Source Metadata



*Define-XML v2 uses 39 data sets

SAS Clinical Standards Toolkit 1.7.1

From Study Source Metadata to Define-XML v2



1

```
%cstutilxmlvalidate();
```

2

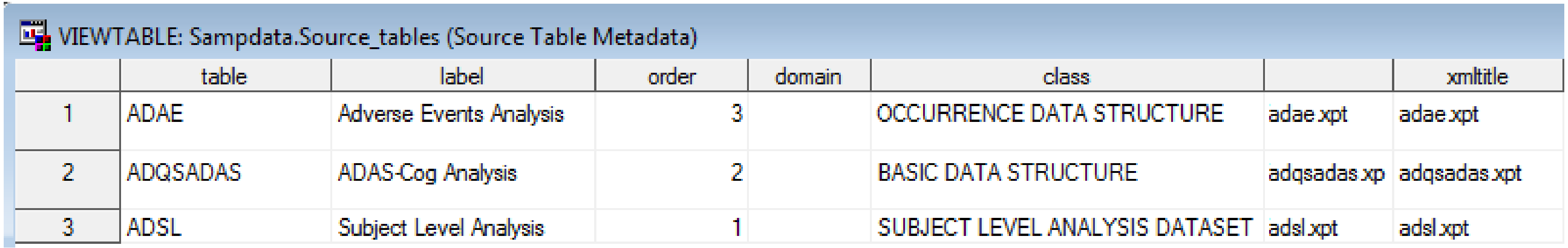
```
%define_read();
```

3

```
%define_createsrcmetafromdefine(
   _cstDefineDataLib=srcdata,
   _cstTrgStandard=CDISC-SDTM,
    cstTrgStandardVersion=3.1.2,
   _cstTrgStudyDS=trgmeta.source_study,
   _cstTrgTableDS=trgmeta.source_tables,
   _cstTrgColumnDS=trgmeta.source_columns,
   _cstTrgCodeListDS=trgmeta.source_codelists,
   _cstTrgValueDS=trgmeta.source_values,
   _cstTrgDocumentDS=trgmeta.source_documents,
    cstTrgAnalysisResultDS=trgmeta.source analysisresults,
   _cstLang=en,
   _cstUseRefLib=Y,
   _cstRefTableDS=refmeta.reference_tables,
   _cstRefColumnDS=refmeta.reference_columns,
   _cstClassTableDS=refmeta.class_tables,
   _cstClassColumnDS=refmeta.class_columns
   );
```

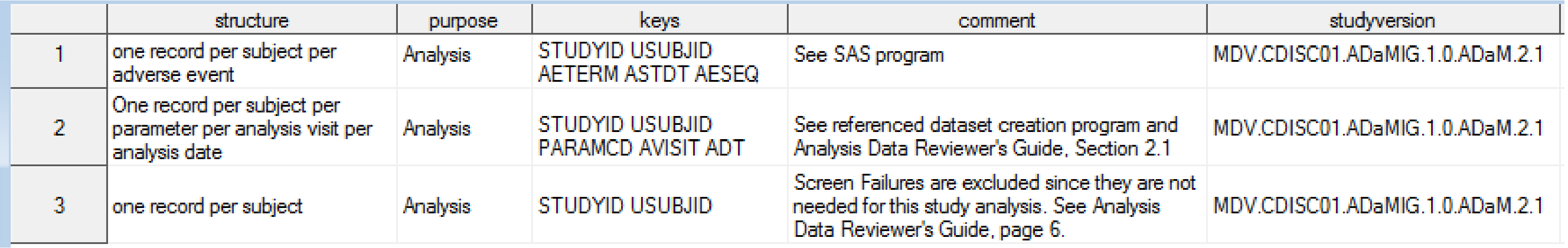
SAS Clinical Standards Toolkit 1.7.1

Source Metadata - source_tables - One record per table



VIEWTABLE: Sampdata.Source_tables (Source Table Metadata)

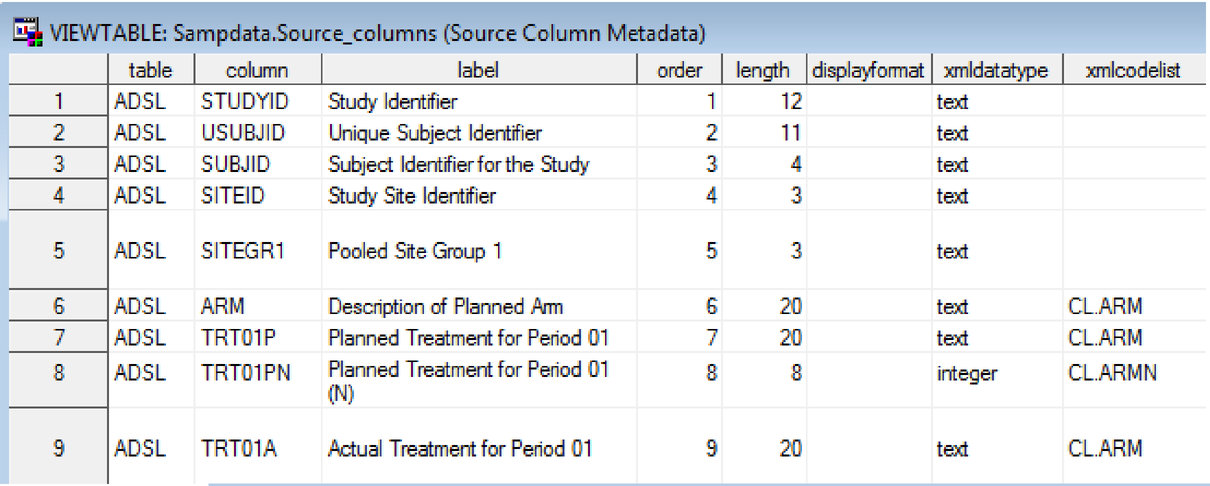
	table	label	order	domain	class		xmltitle
1	ADAE	Adverse Events Analysis	3		OCCURRENCE DATA STRUCTURE	adae.xpt	adae.xpt
2	ADQSADAS	ADAS-Cog Analysis	2		BASIC DATA STRUCTURE	adqsadas.xp	adqsadas.xpt
3	ADSL	Subject Level Analysis	1		SUBJECT LEVEL ANALYSIS DATASET	adsl.xpt	adsl.xpt



	structure	purpose	keys	comment	studyversion
1	one record per subject per adverse event	Analysis	STUDYID USUBJID AETERM ASTDT AESEQ	See SAS program	MDV.CDISC01.ADaMIG.1.0.ADaM.2.1
2	One record per subject per parameter per analysis visit per analysis date	Analysis	STUDYID USUBJID PARAMCD AVISIT ADT	See referenced dataset creation program and Analysis Data Reviewer's Guide, Section 2.1	MDV.CDISC01.ADaMIG.1.0.ADaM.2.1
3	one record per subject	Analysis	STUDYID USUBJID	Screen Failures are excluded since they are not needed for this study analysis. See Analysis Data Reviewer's Guide, page 6.	MDV.CDISC01.ADaMIG.1.0.ADaM.2.1

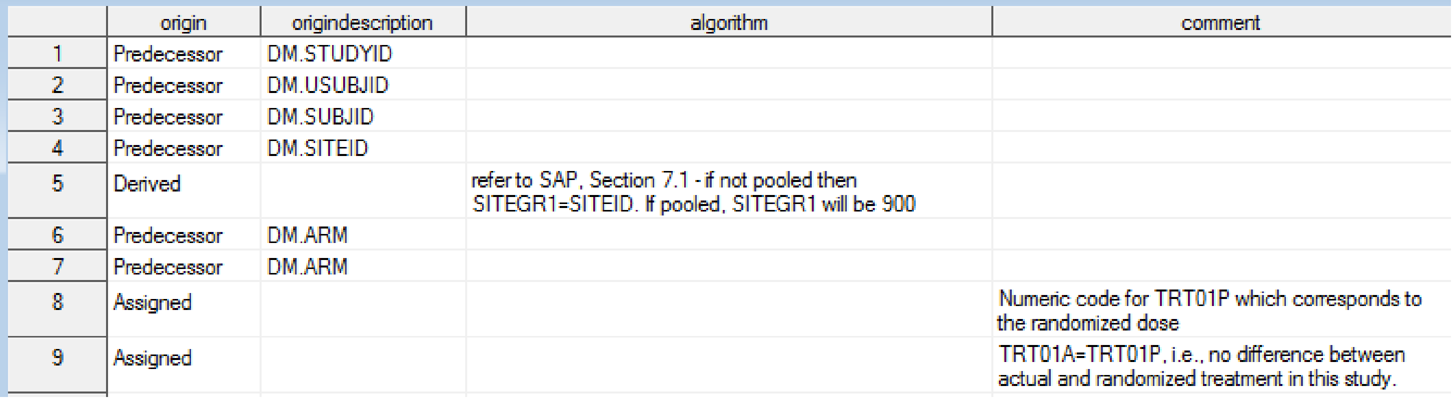
SAS Clinical Standards Toolkit 1.7

Source Metadata - source_columns - One record per table, column



VIEWTABLE: Sampdata.Source_columns (Source Column Metadata)

	table	column	label	order	length	displayformat	xmldatatype	xmlcodelist
1	ADSL	STUDYID	Study Identifier	1	12		text	
2	ADSL	USUBJID	Unique Subject Identifier	2	11		text	
3	ADSL	SUBJID	Subject Identifier for the Study	3	4		text	
4	ADSL	SITEID	Study Site Identifier	4	3		text	
5	ADSL	SITEGR1	Pooled Site Group 1	5	3		text	
6	ADSL	ARM	Description of Planned Arm	6	20		text	CL.ARM
7	ADSL	TRT01P	Planned Treatment for Period 01	7	20		text	CL.ARM
8	ADSL	TRT01PN	Planned Treatment for Period 01 (N)	8	8		integer	CL.ARMN
9	ADSL	TRT01A	Actual Treatment for Period 01	9	20		text	CL.ARM



	origin	origindescription	algorithm	comment
1	Predecessor	DM.STUDYID		
2	Predecessor	DM.USUBJID		
3	Predecessor	DM.SUBJID		
4	Predecessor	DM.SITEID		
5	Derived		refer to SAP, Section 7.1 - if not pooled then SITEGR1=SITEID. If pooled, SITEGR1 will be 900	
6	Predecessor	DM.ARM		
7	Predecessor	DM.ARM		
8	Assigned			Numeric code for TRT01P which corresponds to the randomized dose
9	Assigned			TRT01A=TRT01P, i.e., no difference between actual and randomized treatment in this study.

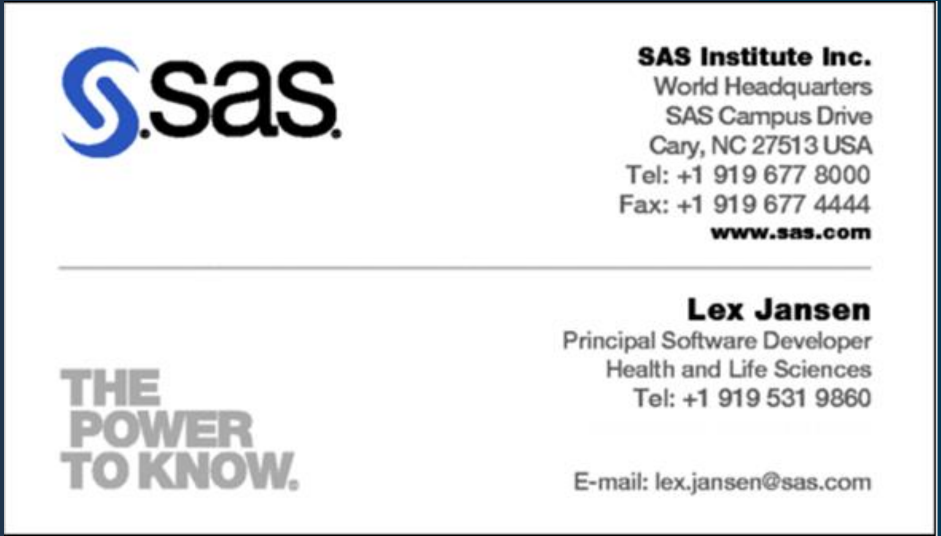
Accessing the Metadata from Define-XML

Conclusion

4 XML aware methods to read Define-XML

- XML Mapper
  - Generic XML tool
  - Can be tedious
  - A lot of post-processing
- PROC GROOVY and PROC XSL
  - New languages means new Skills
  - Extremely flexible and generic XML tools
- SAS Clinical Standards Toolkit
  - Already completely implemented for Define-XML

Thank You !
Questions ?



SAS Institute Inc.
World Headquarters
SAS Campus Drive
Cary, NC 27513 USA
Tel: +1 919 677 8000
Fax: +1 919 677 4444
www.sas.com

Lex Jansen
Principal Software Developer
Health and Life Sciences
Tel: +1 919 531 9860

E-mail: lex.jansen@sas.com

THE POWER TO KNOW.

sas.com