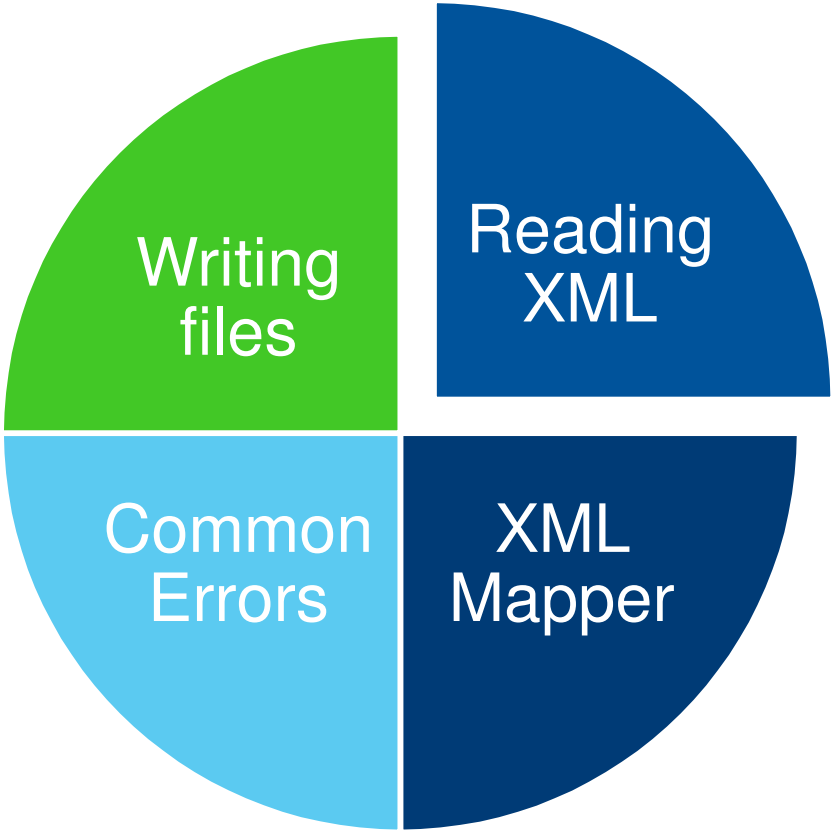
EFFECTIVELY PROCESSING XML USING SAS

CHEVELL PARKER
TECHNICAL SUPPORT ANALYST, SAS INSTITUTE INC.

Overview

- XML basics and Libname engine introduction
- Effectively reading XML files
- Introduction to the XML Mapper
- Writing XML files using SAS
- Common problems processing XML files

Processing XML Files Using SAS



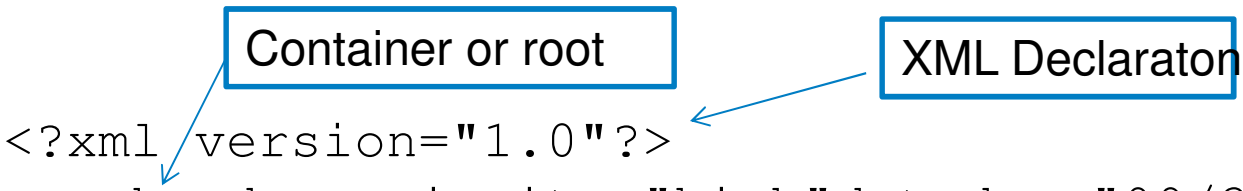
What is XML?

- XML is a set of rules used for defining & modeling structures
- XML is extensible & customizable
  - Its greatest strength
  - Its greatest weakness

XML Basics- Well Formed Files

- Document has a single root element
- Elements nest properly
- No tag omission (close what you open)
- Attributes must be quoted
- Special characters < > and & must always be escaped
- XML is case sensitive

Anatomy of an XML file




```
<?xml version="1.0"?>
<workorder priority="high"datedue="09/30/2001">
  <submitter>
    <name first="Jennifer" last="Kyrnin" />
    <email>html.guide@about.com</email>
    <account number="11001100" />
  </submitter>
  <project title="update aa051198.htm article">
    <url>http://webdesign.com/aa051198.htm</url>
    <description> new article</description>
  </project>
</workorder>
```

Libname Engine Introduction

Release	Engine	Description
SAS 8.1	XML	Export production
SAS 8.2	XML	Ability to read added with hot fix
SAS 9.1	XML	
SAS 9.2	XML|XML92	Enhanced features added to XML92
SAS 9.3 | 9.4	XML|XMLV2	XMLV2 is an alias of XML92

Libname Engine Introduction-Processing XML

- Provides the ability to read and write XML files
- Requires that XML files be well formed for reading
- Reads and writes generic XML files by default

Libname Engine Introduction-Extended Functionality of the XMLV2 Engine

- Wildcards can be used to read all files in a directory
- Allows hierarchical files to be read by dynamically generating Map files
- Namespaces are supported beginning with SAS 9.3
- Provides the ability to use XMLMap files for export as well as import

Effectively Reading XML Files

```
<?xml version="1.0" ?>
<TABLE> ← root node
  <STUDENTS> ← repeating element instance
    <ID> 0755 </ID> ← begin reading row 1
    <NAME> Brad Martin </NAME>
    <ADDRESS> 1611 Glengreen </ADDRESS>
    <CITY> Huntsville </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>

  <STUDENTS> ← repeating element instance
    <ID> 1522 </ID> ← begin reading row 2
    <NAME> Zac Harvell </NAME>
    <ADDRESS> 11900 Glenda </ADDRESS>
    <CITY> Houston </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>
</TABLE>
```

Effectively Reading XML Files

```
Libname test xmlv2 'C:\students.xml';
proc print data=test.students;
run;
```

ID	NAME	ADDRESS	CITY	STATE
0755	Brad Martin	1611 Glengreen	Huntsville	Texas
1522	Zac Harvell	11900 Glenda	Houston	Texas

Effectively Reading XML Files- Attributes Require an XMLMap

```
<?xml version="1.0" ?>
<TABLE>
  <STUDENTS Dept="Math">
    <ID> 0755 </ID>
    <NAME> Brad Martin </NAME>
    <ADDRESS> 1611 Glengreen </ADDRESS>
    <CITY> Huntsville </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>

  <STUDENTS Dept="CSC">
    <ID> 1522 </ID>
    <NAME> Zac Harvell </NAME>
    <ADDRESS> 11900 Glenda </ADDRESS>
    <CITY> Houston </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>
</TABLE>
```

Attribute

Attribute

Effectively Reading XML Files-Attributes Require an XMLMap

```
Libname test xmlv2 'C:\students.xml';
proc print data=test.students;
run;
```

ID	NAME	ADDRESS	CITY	STATE
0755	Brad Martin	1611 Glengreen	Huntsville	Texas
1522	Zac Harvell	11900 Glenda	Houston	Texas

Effectively Reading XML Files-Required Structure

Problem.XML

<xml version="1.0" ?>	Declaration statement
<Table>	SAS recognizes the first instance tag as the root enclosing element, which is the document container.
<Students>	Starting with the second-level instance tag, SAS begins to scan for columns.
<Student>	SAS expects to start reading data here and because there is no data an error occurs
<ID> 0755 </ID> <NAME> Brad Martin</NAME> <ADDRESS> 1611 Glengreen </ADDRESS> <CITY> Huntsville </CITY> <STATE> Texas </STATE> </Student> </Students> </Table>	

Effectively Reading XML Files-Required Structure

```
libname temp xmlv2 'c:\problem.xml';

proc copy in=temp out=work;
run;
```

```
ERROR: XML data is not in a format supported natively by the XML libname
engine.Files of this
      type may require an XMLMap to be input properly.
NOTE: Statements not processed because of errors noted above.
NOTE: PROCEDURE COPY used (Total process time):
      real time          0.10 seconds
      cpu time           0.01 seconds
```

Effectively Reading XML Files-XMLMap Files

- An XML file that describes XML markup to the engine
- Allows meta data to be added such as the data type, formats and informats, lengths and more
- Element names can be modified making them legal for SAS
- XMLMap files can be created a variety of ways
- Maps function much like the industry standard XML schema

Effectively Reading XML Files- XMLMap Files and XPATH Syntax

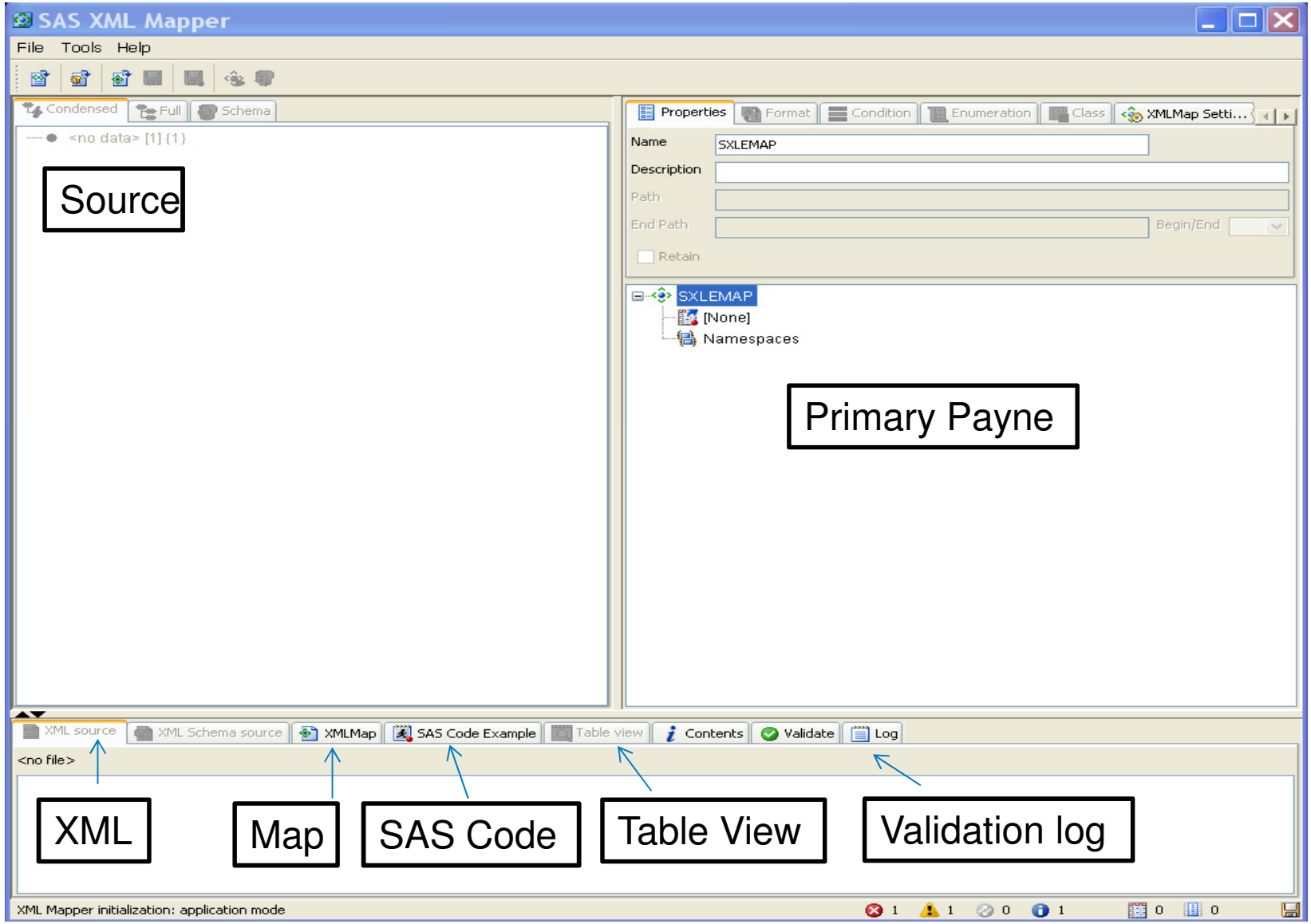
Untitled - Notepad

File Edit Format View Help

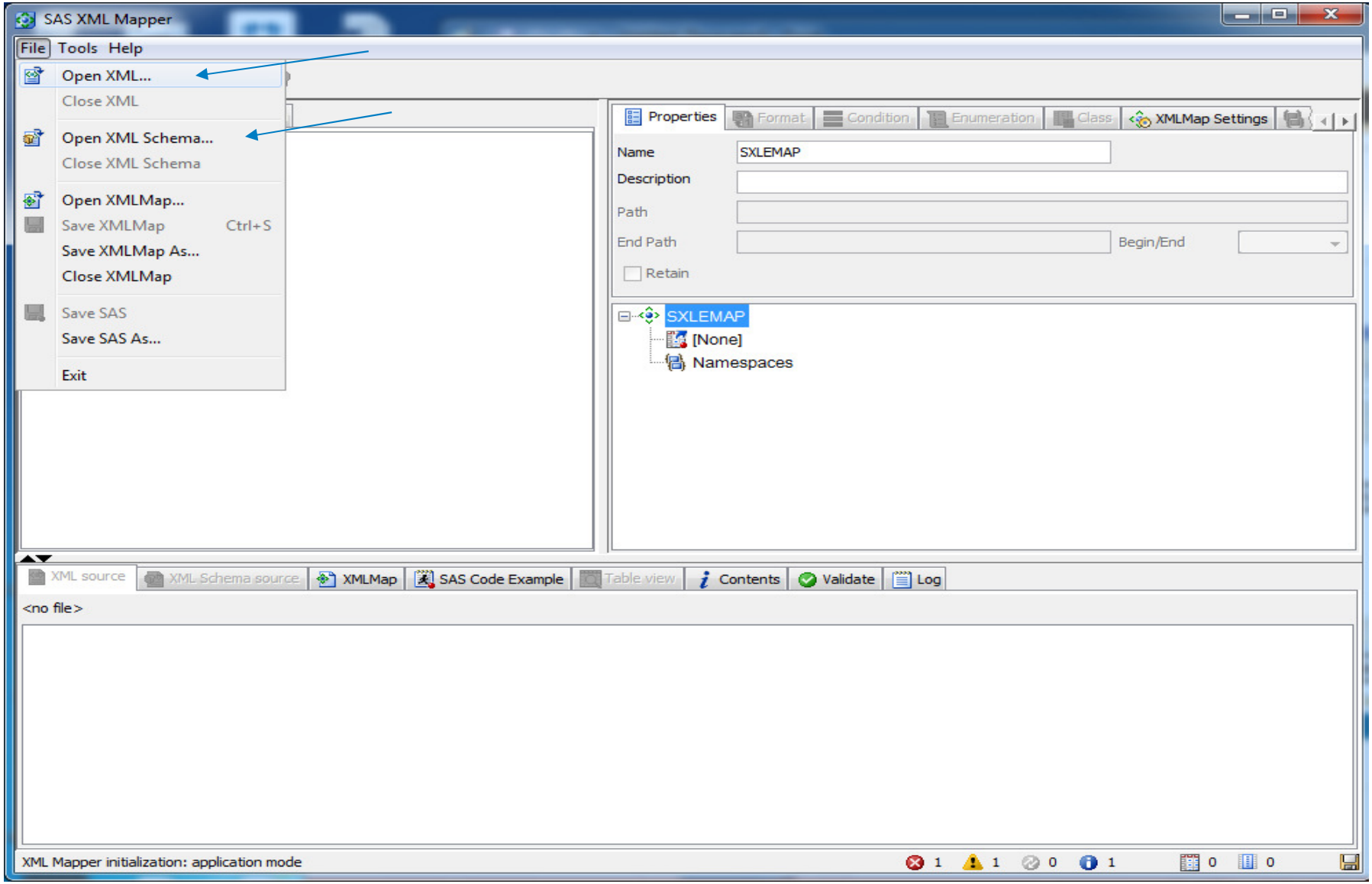
```
?xml version="1.0" encoding="windows-1252"?>
<!-- ############################################################ -->
<!-- 2012-11-29T21:26:29 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 903200.3.0.20120523190000_v930m2 -->
<!-- ############################################################ -->
<SXLEMAP name="AUTO_GEN" version="2.1">

    <NAMESPACES count="0"/>

    <!-- ############################################################ -->
    <TABLE description="STUDENTS" name="STUDENTS">
        <TABLE-PATH syntax="XPath">/LIBRARY/STUDENTS</TABLE-PATH>

       <COLUMN name="ID">
            <PATH syntax="XPath">/LIBRARY/STUDENTS/ID</PATH>
            <TYPE>numeric</TYPE>
            <DATATYPE>integer</DATATYPE>
        </COLUMN>

        <COLUMN name="NAME">
            <PATH syntax="XPath">/LIBRARY/STUDENTS/NAME</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>11</LENGTH>
        </COLUMN>

        <COLUMN name="ADDRESS">
            <PATH syntax="XPath">/LIBRARY/STUDENTS/ADDRESS</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>14</LENGTH>
        </COLUMN>

        <COLUMN name="CITY">
            <PATH syntax="XPath">/LIBRARY/STUDENTS/CITY</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>10</LENGTH>
        </COLUMN>

        <COLUMN name="STATE">
            <PATH syntax="XPath">/LIBRARY/STUDENTS/STATE</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>5</LENGTH>
        </COLUMN>

    </TABLE>
```

```
<?xml version="1.0" ?>
<LIBRARY>
  <STUDENTS>
     <ID> 0755 </ID>
     <NAME> Brad Martin </NAME>
     <ADDRESS> 1611 Glengreen
      </ADDRESS>
     <CITY> Huntsville </CITY>
     <STATE> Texas </STATE>
  </STUDENTS>
</LIBRARY>
```

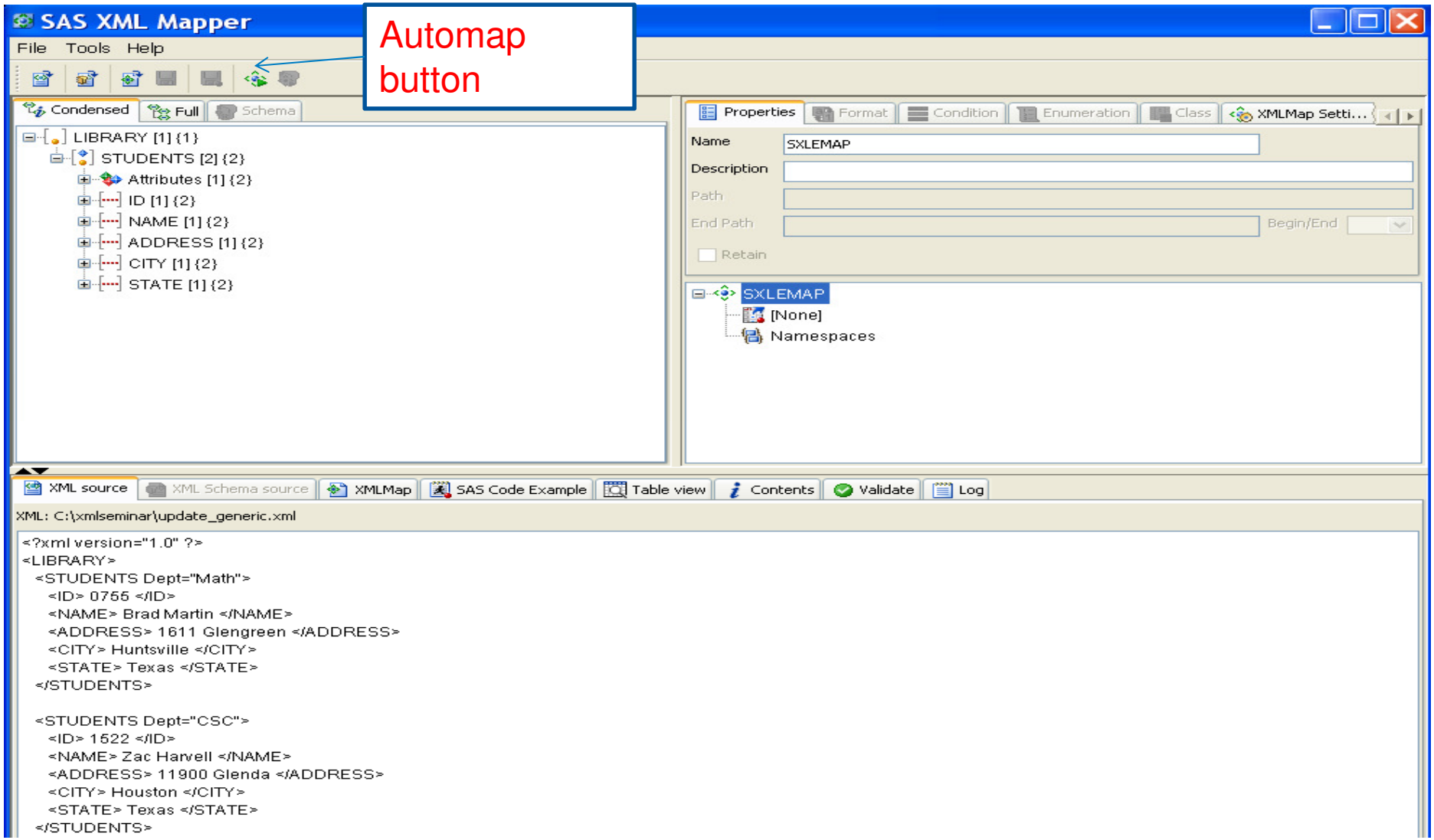
The XML Mapper



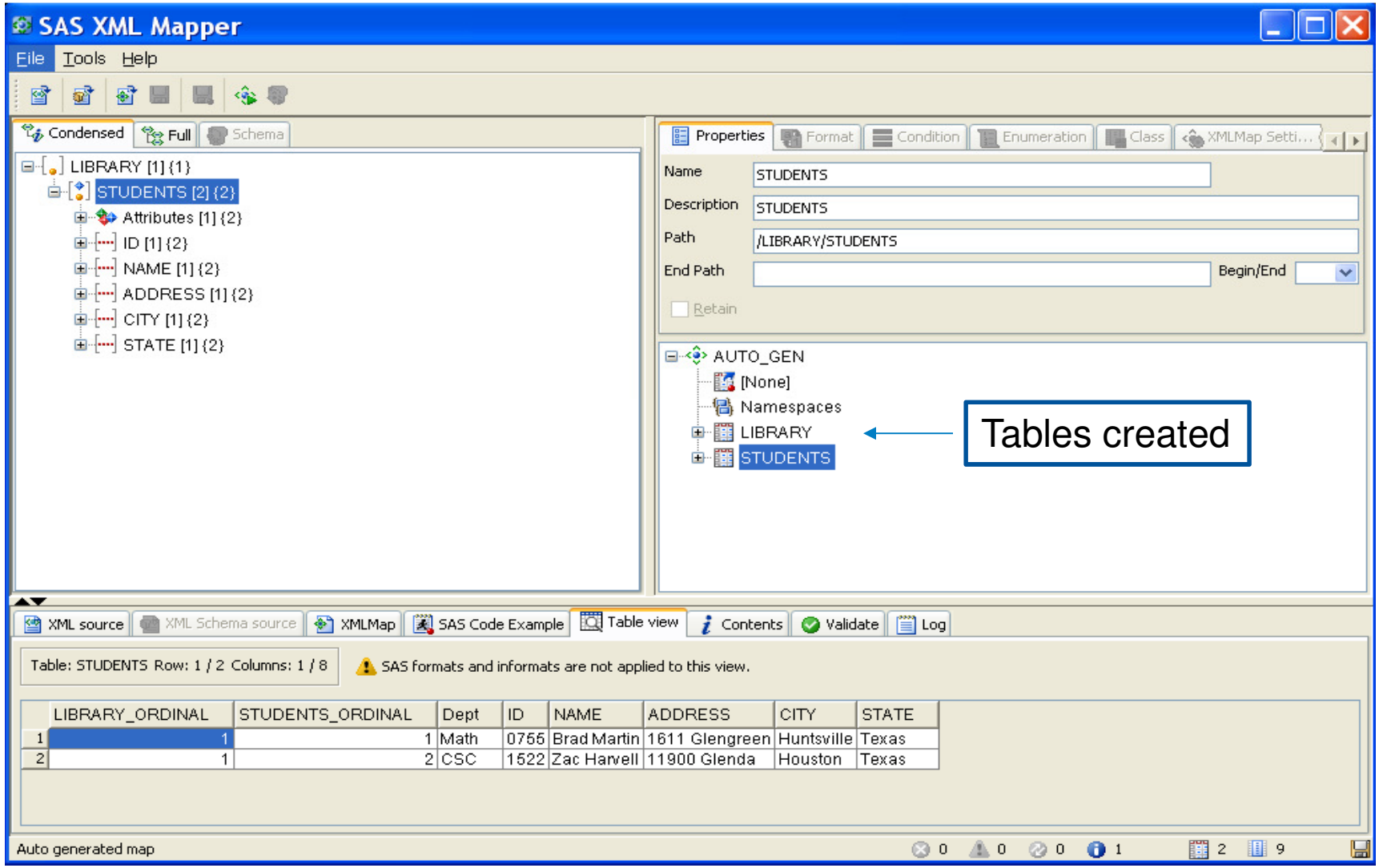
The XML Mapper – Loading Files to Map



The XML Mapper – AUTOMAP Feature



The XML Mapper – AUTOMAP Feature



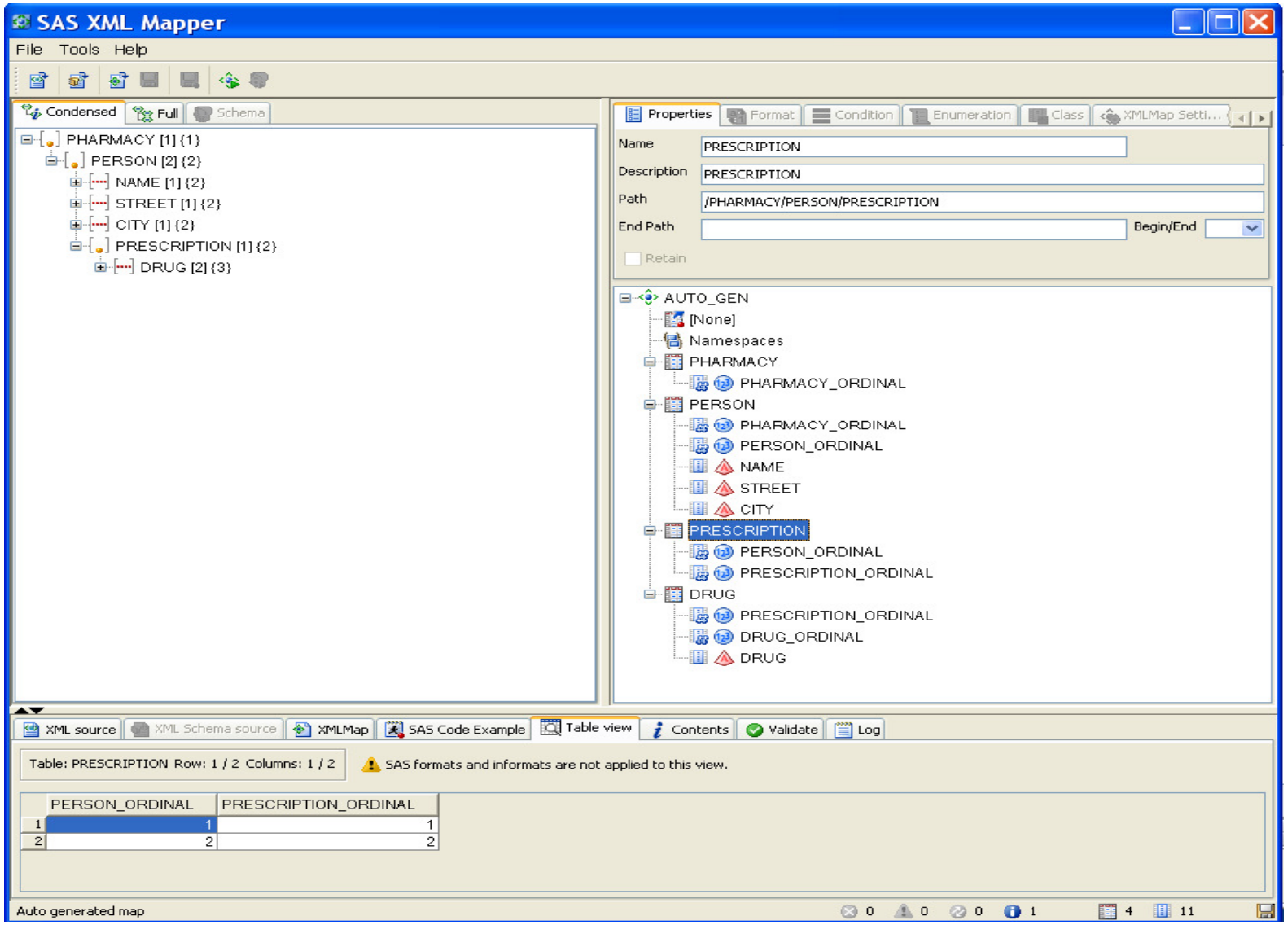
XML Mapper-AUTOMAP

- Generates one or more tables depending on file structure
- Creates ordinal columns which acts as a keys by default
- Character values use length of largest value
- Can generate map files from a Schema or XML data files

XML Mapper- AUTOMAP and Table Generation

```
<?xml version="1.0" ?>
<PHARMACY>    (1)
 <PERSON>      (2)
   <NAME>Brad Martin</NAME>
   <STREET>11900 Glenda Court</STREET>
   <CITY>Austin</CITY>
   <PRESCRIPTION>   (3)
      <DRUG>Tetracycline</DRUG>  (4)
      <DRUG>Lomotil</DRUG>
   </PRESCRIPTION>
 </PERSON>
 <PERSON>
   <NAME>Jim Spano</NAME>
   <STREET>1611 Glengreen</STREET>
   <CITY>Austin</CITY>
   <PRESCRIPTION>
       <DRUG>Nexium</DRUG>
   </PRESCRIPTION>
 </PERSON>
</PHARMACY>
```

XML Mapper- AUTOMAP and Table Generation



XML Mapper Icons

Elements

Empty element

Element with attributes

Element with child elements

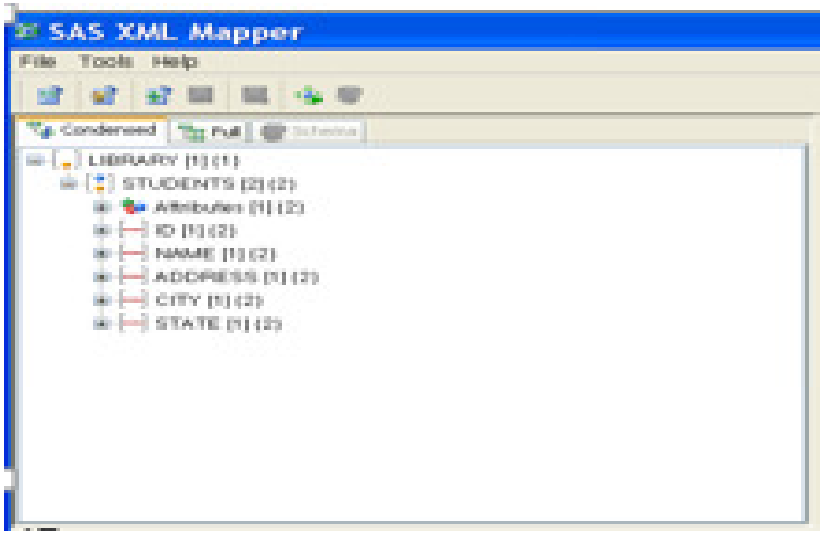
Element with data

Element with both attributes and child elements

Element with both data and attributes

Element with both data and child elements

Element with data, attributes, and child elements



The XML Mapper – Creating Custom Map Files

- Create the correct table boundary on the file
- Avoiding truncated records
- Create custom ordinal by modifying path
- Changing formats and meta data
- Other advanced features

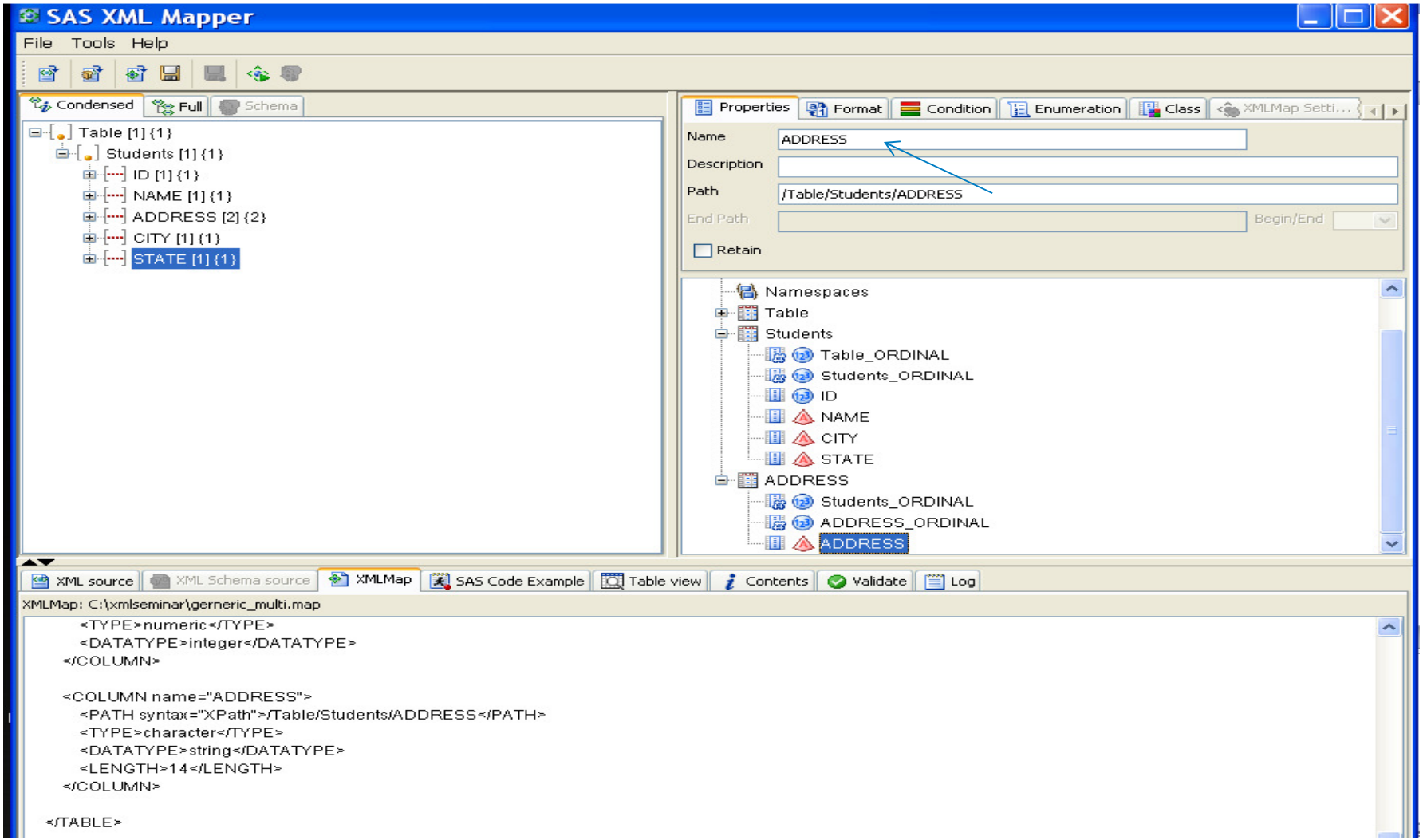
The XML Mapper – Custom Maps and Combining a Tables

```
filename  person 'C:\person.xml';
filename  SXLEMAP 'C:\person.map';
libname   person xmlv2 xmlmap=SXLEMAP ;

data prescription;
   merge person.PRESCRIPTION person.DRUG;
   by prescription_ordinal;
run;

data person;
  merge  person.person prescription;
  by person_ordinal;
 run;

proc print;
run;
```

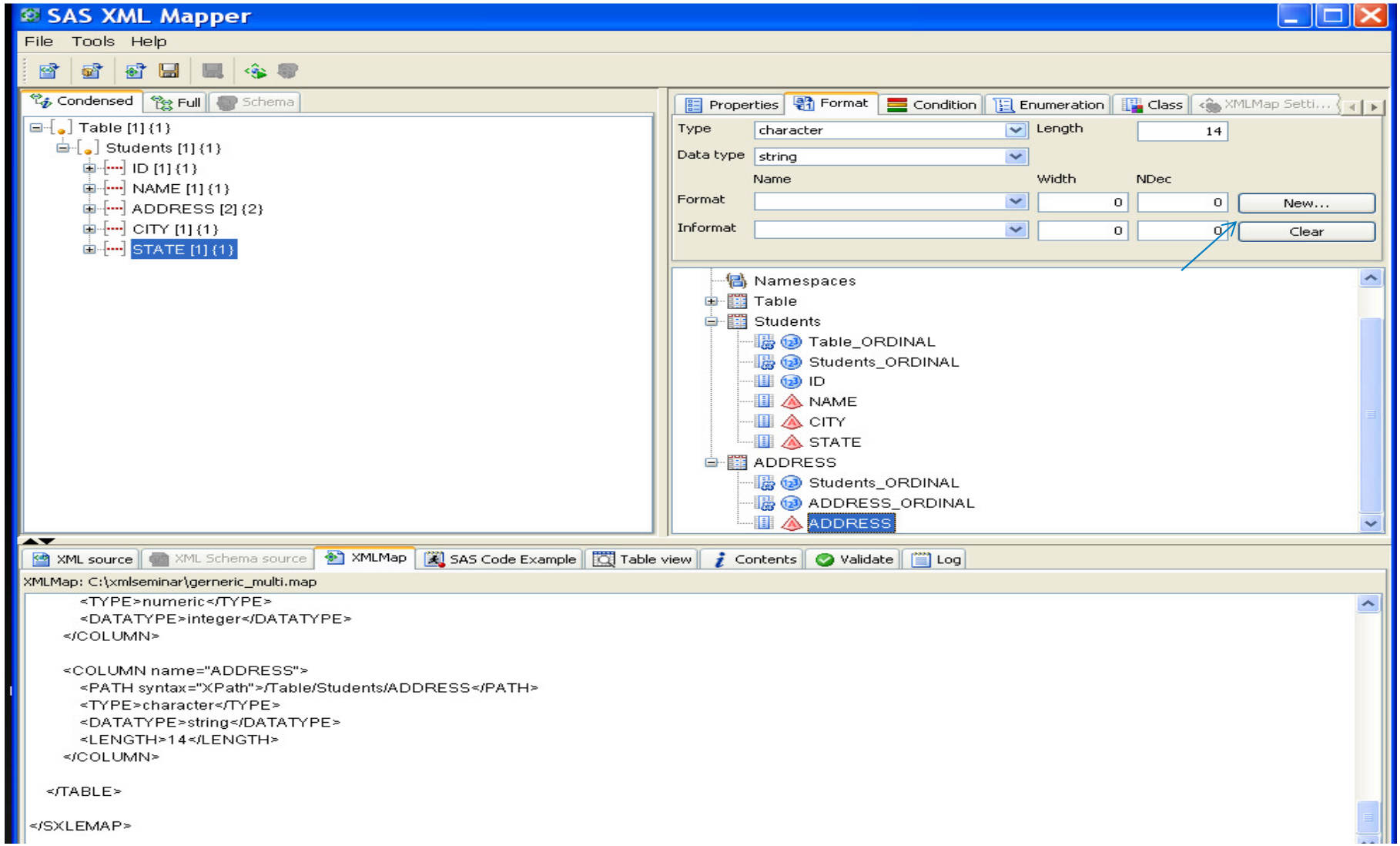
The XML Mapper – Custom Maps and Combining a Tables

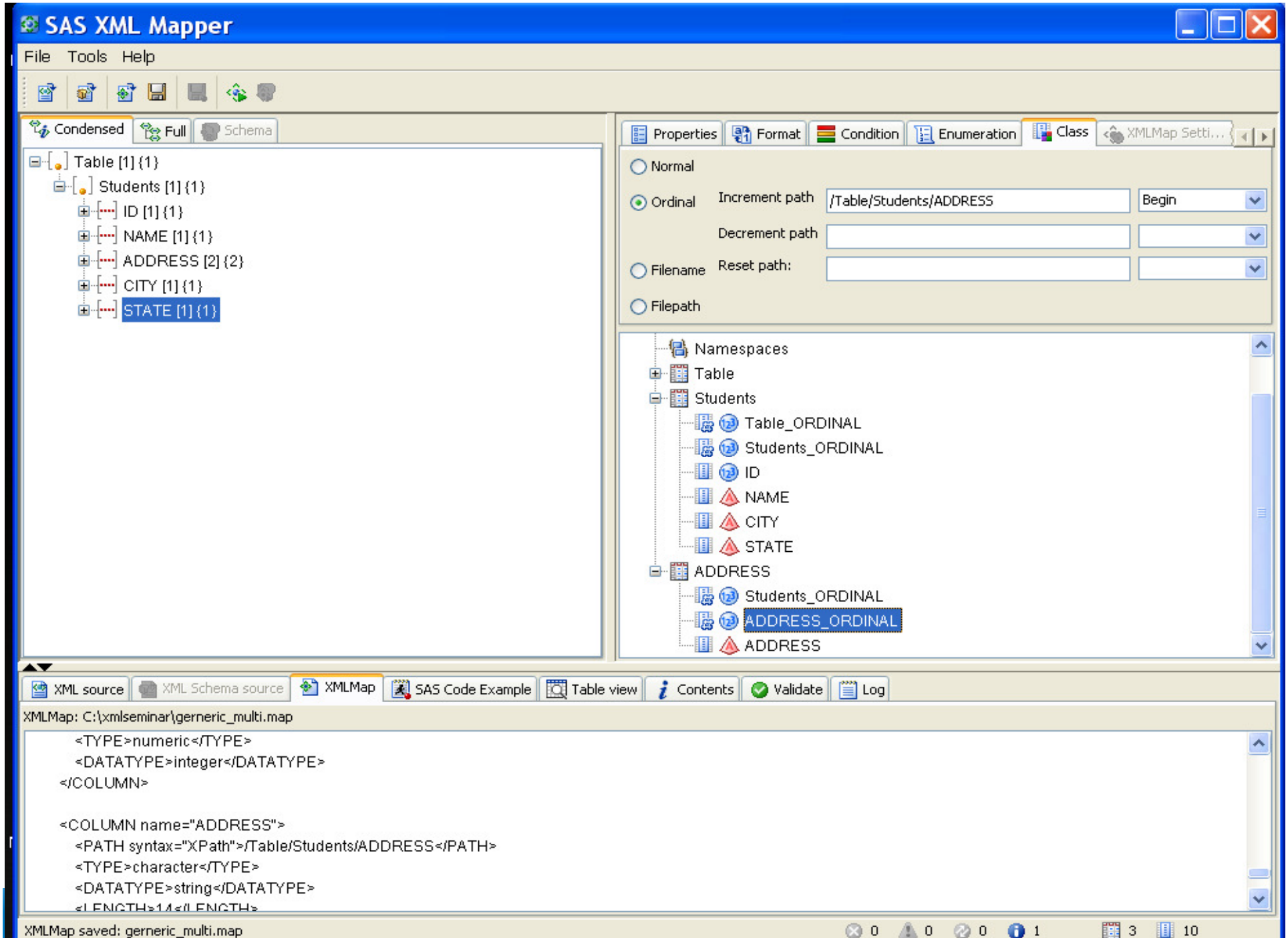
Obs	PHARMACY_ ORDINAL	PERSON_ ORDINAL	NAME	STREET	CITY	PRESCRIPTION_ ORDINAL	DRUG_ ORDINAL	DRUG
1	1	1	Brad Martin	11900 Glenda Court	Austin	1	1	Tetracycline
2	1	1	Brad Martin	11900 Glenda Court	Austin	1	2	Lomotil
3	1	2	Jim Spano	1611 Glengreen	Austin	2	3	Nexium

The XML Mapper – Modifying Column Names and Formats



The XML Mapper – Modifying Column Names and Formats




SAS XML Mapper
File Tools Help
Condensed Full Schema
Table [1] {1}
Students [1] {1}
ID [1] {1}
NAME [1] {1}
ADDRESS [2] {2}
CITY [1] {1}
STATE [1] {1}
Properties Format Condition Enumeration Class XMLMap Setti...
Normal
Ordinal
Filename
Filepath
Increment path
/Table/Students/ADDRESS
Begin
Decrement path
Reset path:
Namespaces
Table
Students
Table_ORDINAL
Students_ORDINAL
ID
NAME
CITY
STATE
ADDRESS
Students_ORDINAL
ADDRESS_ORDINAL
ADDRESS
XML source XML Schema source XMLMap SAS Code Example Table view Contents Validate Log
XMLMap: C:\xmlseminar\gerneric_multi.map
<TYPE>numeric</TYPE>
<DATATYPE>integer</DATATYPE>
</COLUMN>
<COLUMN name="ADDRESS">
<PATH syntax="XPath">/Table/Students/ADDRESS</PATH>
<TYPE>character</TYPE>
<DATATYPE>string</DATATYPE>
XMLMap saved: gerneric_multi.map
0 0 0 1 3 10

Dynamically Generating XMLMap files from SAS Beginning with Version 9.3M2

```
filename data 'c:₩example.xml';
filename map  'c:₩example.map';

libname data xmlv2 xmlmap=map automap=replace;

proc copy in=data out=work;
run;
```

Exporting XML Files using the XMLV2 Engine

Generic

```
libname temp xmlv2 'c:₩example.xml';

data temp.class;
    set sashelp.class;
run;
```

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
- <TABLE>
  - <CLASS>
      <Name>Joyce</Name>
      <Sex>F</Sex>
      <Age>11</Age>
      <Height>51.3</Height>
      <Weight>50.5</Weight>
    </CLASS>
  - <CLASS>
      <Name>Thomas</Name>
      <Sex>M</Sex>
      <Age>11</Age>
      <Height>57.5</Height>
      <Weight>85</Weight>
    </CLASS>
  - <CLASS>
      <Name>James</Name>
      <Sex>M</Sex>
      <Age>12</Age>
      <Height>57.3</Height>
      <Weight>83</Weight>
    </CLASS>
  - <CLASS>
      <Name>Jane</Name>
      <Sex>F</Sex>
      <Age>12</Age>
      <Height>59.8</Height>
      <Weight>84.5</Weight>
```

Exporting XML Files using the XMLV2 Engine

```
libname temp xmlv2 'c:₩example.xml ‘ xmltype=oracle;

data temp.class;
    set sashelp.class;
run;
```

Oracle

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
- <ROWSET>
  - <ROW>
      <Name> Joyce </Name>
      <Sex> F </Sex>
      <Age> 11 </Age>
      <Height> 51.3 </Height>
      <Weight> 50.5 </Weight>
    </ROW>
  - <ROW>
      <Name> Thomas </Name>
      <Sex> M </Sex>
      <Age> 11 </Age>
      <Height> 57.5 </Height>
      <Weight> 85 </Weight>
    </ROW>
  - <ROW>
      <Name> James </Name>
      <Sex> M </Sex>
      <Age> 12 </Age>
      <Height> 57.3 </Height>
      <Weight> 83 </Weight>
    </ROW>
  - <ROW>
      <Name> Jane </Name>
      <Sex> F </Sex>
      <Age> 12 </Age>
      <Height> 59.8 </Height>
      <Weight> 84.5 </Weight>
```

Exporting XML Files using the XMLV2 Engine-Tagsets

- Custom tagsets can be generated to control the layout of export XML file
- Other shipped tagsets can be used to control the tagging structure such as how missing values are displayed
- The SASXMOG tagset is the default tagset used by the XML and XMLV2 engines

Common Questions –Where can I get a copy of the XML mapper?

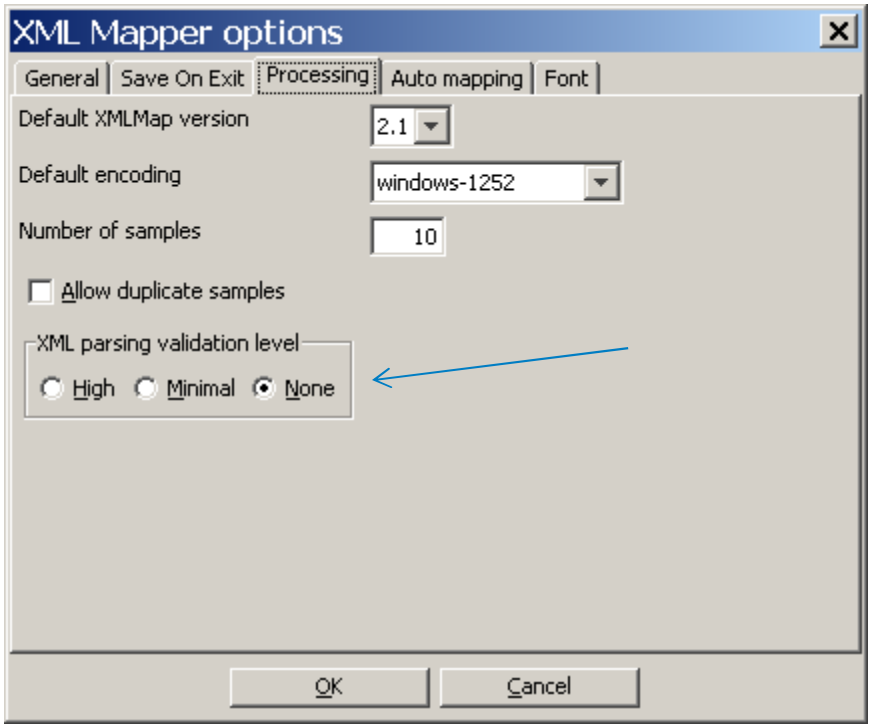
- Install it from the demos and download site located at support.sas.com web site or SAS Media
- It's a stand alone Java application so it needs to follow the JRE guidelines
- If SAS versions prior to SAS 9.3 should use SAS 9.2 version of the Mapper

Common Questions –I am getting an Out of Memory Error using the XML mapper

- Create a map from a Schema (XSD) file rather than the XML data file
- Map a simple representation from the file rather than the entire file
- Increase the heap space for the JRE using the -XMX and –XMS options

Common Questions – Why am I getting Validate Errors Loading XML Files in the Mapper?

Modify parsing validation level



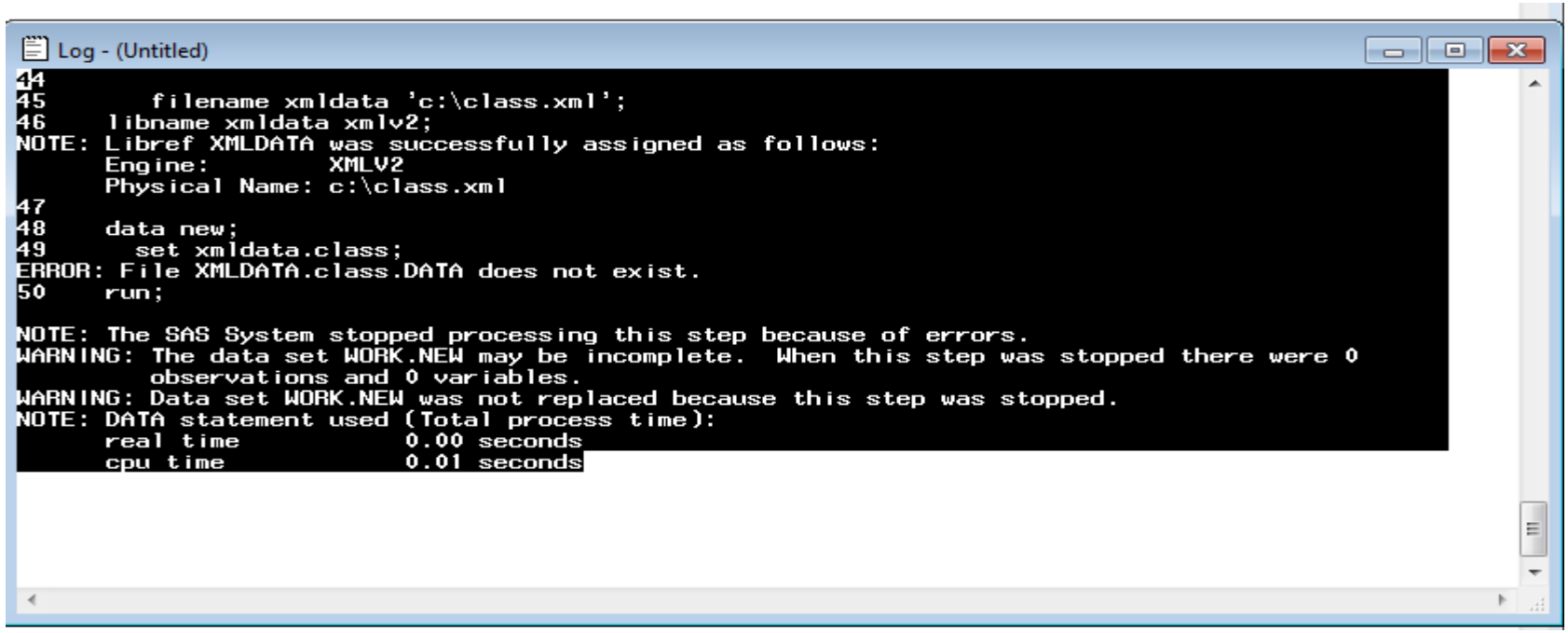
Common Questions –Why am I getting Transcode Errors Using the XMLV2 Engine

- Characters in the file cannot be represented correctly with the current session encoding
- They were ignored with the XML engine and not the XMLV2 engine
- Future option may be implemented to restore the behavior of the XML Engine

Common Questions –Why am I getting the File does not exist Error?

```
filename xmldata 'c:₩example.xml';
libname xmldata xmlv2;

data new;
  set xmldata.example;
run;
```

Verify that this is a valid table by the XMLv2 Engine



Resources

Base Engine
http://support.sas.com/rnd/base/xmlengine/index.html

XML Engine Tip Sheet
http://support.sas.com/rnd/base/xmlengine/XML94tipsheet.pdf

Contact

Email: Chevell.Parker@sas.com