

The “Ins” and “Outs” of XML

Reading and writing XML documents
using SAS

Agenda

- Overview of XML
- Creating SAS Data Sets from XML documents (“Ins”)
- Creating XML documents from SAS Data Sets (“Outs”)
- Q & A

Overview of XML

- What is XML?
- Benefits of XML
- XML vs HTML
- XML Structure
- Practical Example

What is XML?

Extensible Markup Language

- Custom tag sets
- Used to **describe** data
- Well suited to **data exchange**
- Contents of XML is **text**

“XML is to data applications
what ASCII is to Word Processors”

Benefits of XML

- Flexible tag library
- Platform Independent
- Mechanism for exchanging unformatted data
- Validation routines to ensure integrity of data

How is HTML Different from XML?

HTML

HTML (Code)

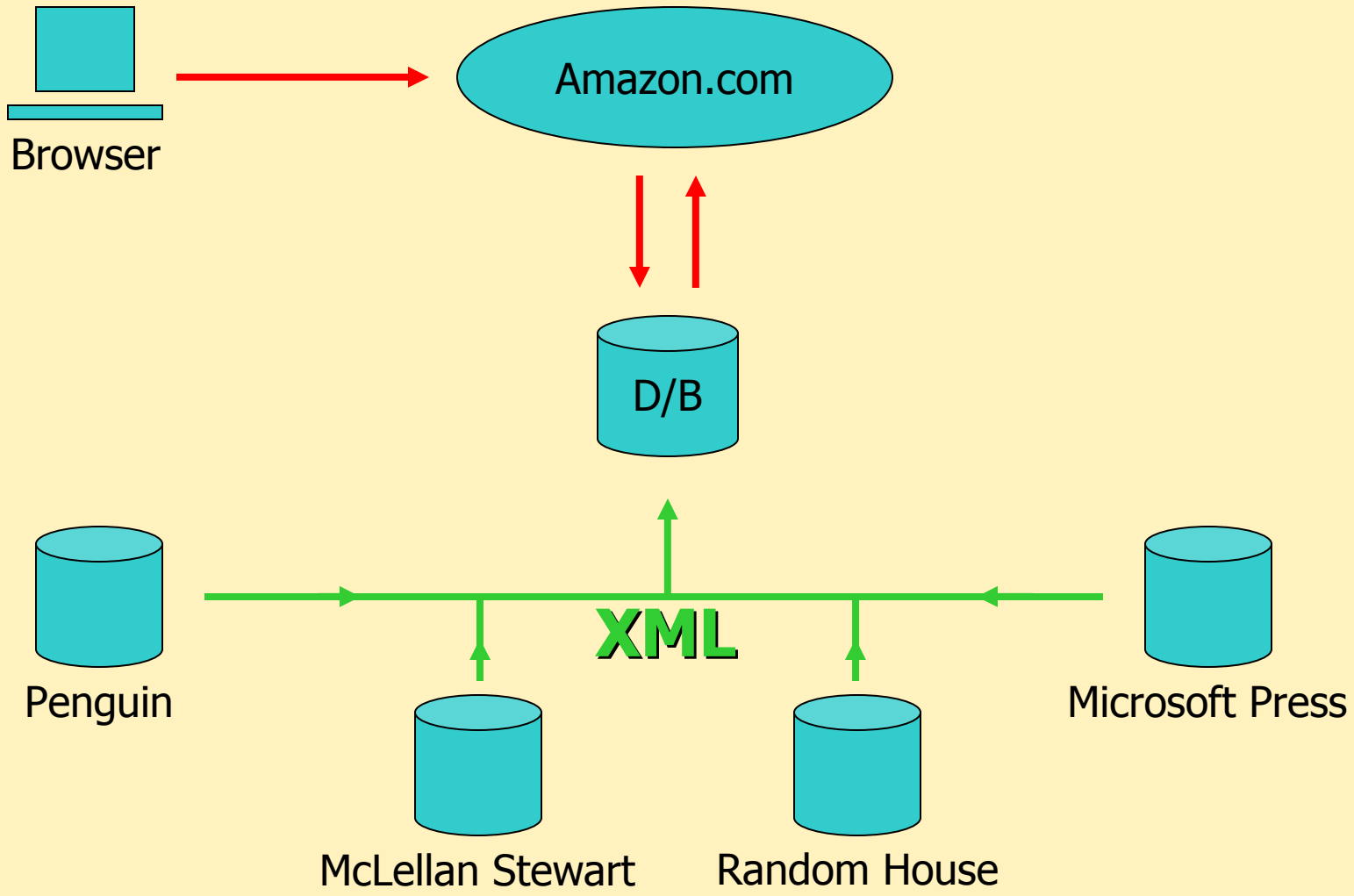
XML

XML Structure

- Extremely Rigid Syntax
 - Case sensitive
 - Requires Containment
- Structure enforced by XML spec
 - Defined by a Standards organization
- Data Integrity enforced by DTD or XSD
 - Created and shared between parties
 - Entire document **MUST** be valid or completely ignored

DTD = Document Type Definition

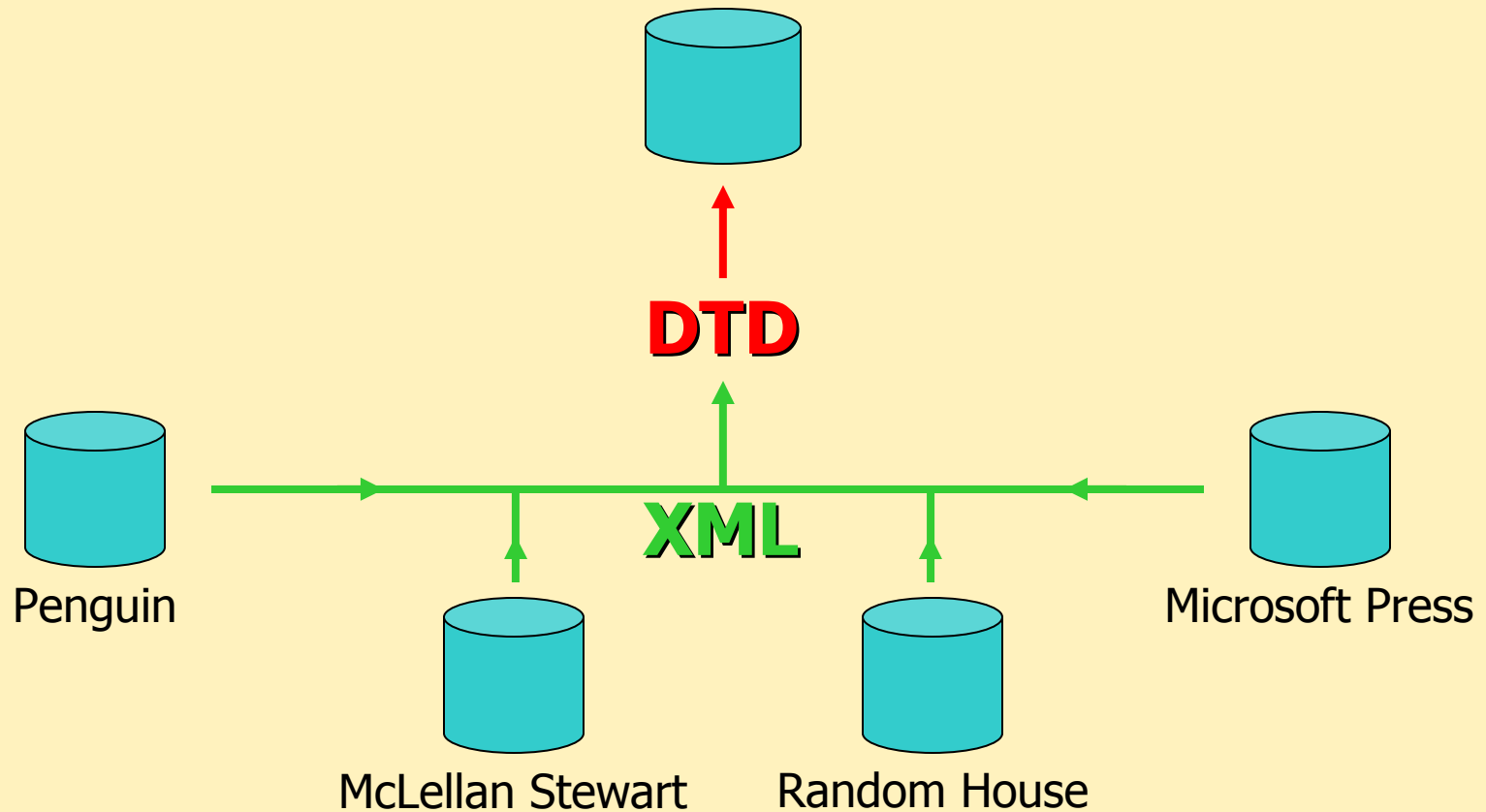
XSD = XML Schema Definition



```
<!ELEMENT Book>
<!ELEMENT Title CDATA #REQUIRED>
<!ATTLIST ISBN CDATA #REQUIRED>
<!ELEMENT Author CDATA #REQUIRED>
<!ELEMENT Format CDATA #REQUIRED>
<!ELEMENT Genre CDATA #REQUIRED>
<!ELEMENT Cost (#PCDATA) #REQUIRED>
<!ELEMENT CoverImage>
```

Things that might not be in DTD ...

List Price
Book Club Price
Editors Notes
Promotion Code
Etc.

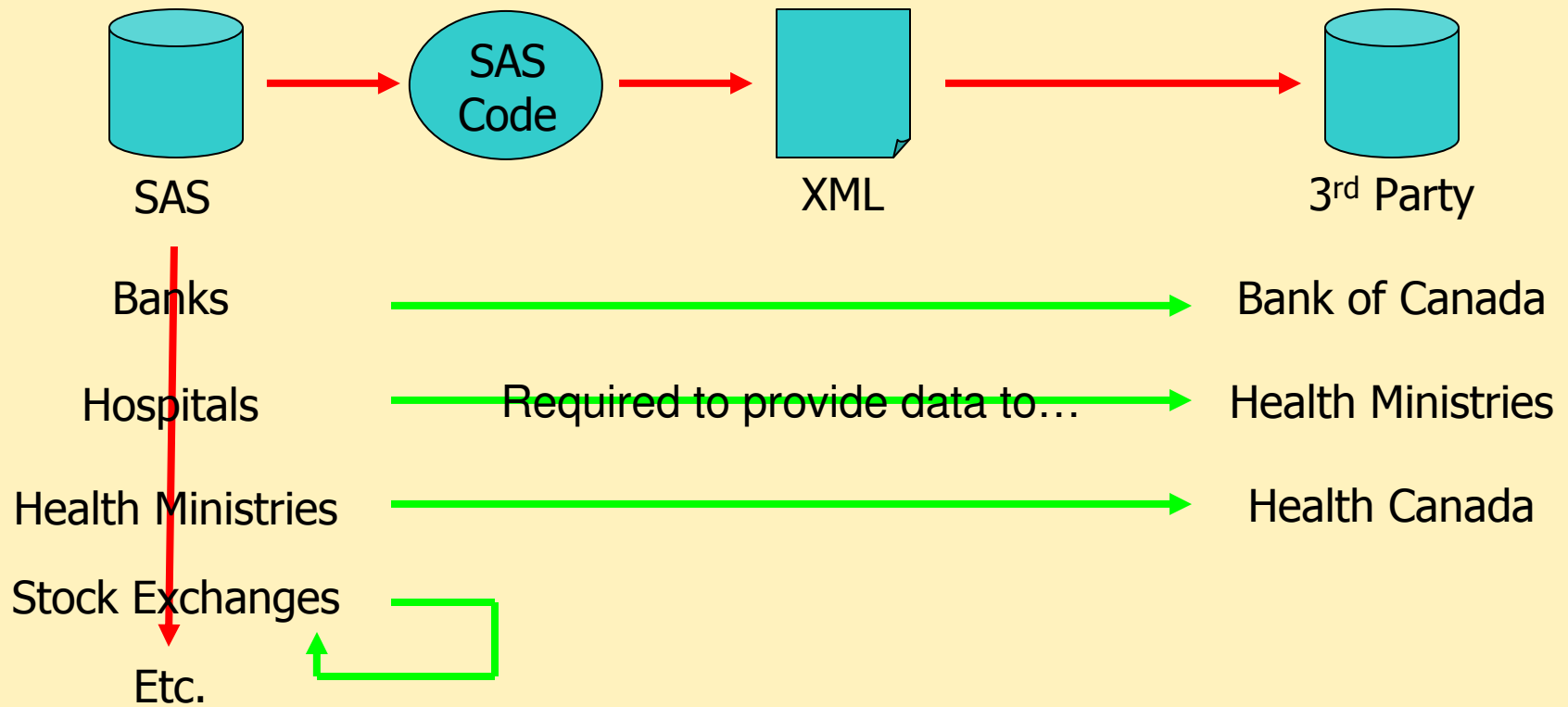


Benefits of XML - Review

- Flexible tag library
- Platform Independent
- Mechanism for exchanging unformatted data
- Validation routines to ensure integrity of data

All make XML an increasingly popular and viable option for data exchange

XML & SAS



Creating SAS Data Sets from XML documents

- Demo 1
 - Using the SAS XML Libname engine

- Demo 2
 - Using an XMLMap to parse XML files

Creating XML documents from SAS Data Sets

- Demo 3
 - Output default XML documents

- Demo 4
 - Create custom XML document (XML Libname)
 - » Change names and remove data

- Demo 5
 - Create custom XML document (Manual)
 - » Change structure, change names and remove data

Q & A

General Questions?

Solution: Demo 1

Scenario:

3rd party provides you with XML.

They have taken care to provide data in the structure you have defined for them.

```
/* Invoke the XML Libname engine by denoting "XML" as the type */  
libname xmlin xml 'C:\Input.xml';
```

```
/* create a SAS Data Set called "ShortBookList" */  
data xmlbooks.shortbooklist;
```

```
/* Reference the first child node ("book") of the root based on the  
contents of the XML document. All elements contained within this node  
will be used to create corresponding variables */  
set xmlin.book;
```

```
run;
```

Solution: Demo 2

Scenario:

3rd party provides you with XML.

The data is not in a structure you have defined, and you need a different structure in your data set.

/ Create an XMLMAP, then direct the XML Libname engine to the MAP when reading the file */*

```
filename BookFix 'C:\Input.xml';
```

```
filename MAP 'C:\XMLMap.xml';
```

```
libname BookFix xml xmlmap=MAP;
```

```
data xmlbooks.BookListByCategory;
```

```
    set BookFix.Books;
```

```
run;
```

Sample XMLMap: Demo 2

```
<?xml version="1.0" ?>
<SXLEMAP version="1.2">
  <TABLE name="Books">
    <TABLE-PATH syntax="xpath">
      /BookList/Category/Book
    </TABLE-PATH>

    <COLUMN NAME="Title" retain="YES">
      <PATH>
        /BookList/Category/Book/Title
      </PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
    </COLUMN>

    <COLUMN NAME="Author" retain="YES">
      <PATH>
        /BookList/Category/Book/Author
      </PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
    </COLUMN>

    <COLUMN NAME="ISBN">
      <PATH>
        /BookList/Category/Book/Title/@ISBN
      </PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>10</LENGTH>
    </COLUMN>

    <COLUMN NAME="Cost">
      <PATH>
        /BookList/Category/Book/Cost
      </PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>double</DATATYPE>
      <FORMAT width="8" ndec="2">dollar</FORMAT>
    </COLUMN>
  </TABLE>
</SXLEMAP>
```

Solution: Demo 3

Scenario:

3rd party needs your data.

They will use an XML utility to rename nodes and choose only those data items they require, so you can output your data "as-is".

```
/*          Invoke the XML Libname engine by denoting "XML" as the type          */  
libname xmlout xml 'C:\Output.xml';  
  
data xmlout.booklist;  
    set books.booklist;  
  
run;
```

Solution: Demo 4

Scenario:

3rd party needs your data.

They want Author renamed "PennedBy".

They do not want/need Description.

```
/*          create a new temporary data set and use the rename option in the
           set statement to rename required variables          */
data temp.CustomXMLTags;
           set xmlbooks.booklist(rename=(Author=PennedBy) drop=Description);
run;

libname xmlout xml 'C:\Output.xml';

data xmlout.booklist;
           set temp.CustomXMLTags;
run;
```

Solution: Demo 5 (Part 1 of 3)

Scenario:

3rd party needs your data.

Root Node must be called "Inventory".

Book data must be in an Element called "BookTitle" and grouped by an Element called "Category".

Genre must be an attribute of the BookTitle Element.

Author must be "PennedBy"

Description and TimesRead are not to be included

```
/*      The XML engine accounts for ampersand replacement in XML, but we are not
        using the XML engine when building the file manually. Therefore, we will
        use a macro that will search any obseration value it is sent for any
        ampersands and replace each with &amp; - the ANSI value for ampersand      */
```

```
%MACRO fixAmpersand(obtofix);
    ReplaceSearch = prxparse("s/&/&amp;");
    call prxchange(ReplaceSearch, -1, &obtofix);
%MEND;
```

```
/*      Temporarily sort the data set so we can group on the Category variable and
        create different "nodes" for each unique Category      */
```

```
proc sort data=xmlbooks.booklist out=temp.booklist;
    by Category;
run;
```

Solution: Demo 5 (Part 2 of 3)

```
/*           We only need a file as an output, not a new data set, so we perform the
              data step with _null_ as the target data set          */
data _null_;
    set temp.booklist end=LastRecord;
    by Category;
    file 'C:\Output.xml';

/*           if reading the very first record, ensure that the XML definition is
              inserted into the XML document at the top          */
    if _n_ = 1 then do;
        put '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
        put '<Inventory>';
    end;

/*           for each new category, create a new Category node          */
    if First.Category then do;
        put '<Category>';
        put Category;
    end;
```

Continued on next slide

Solution: Demo 5 (Part 3 of 3)

```
put '<Book>';
    Title_Genre = '<BookTitle Genre="" || Genre || "">';
    put Title_Genre;
    %fixAmpersand(Title);
    put Title;
    put '</BookTitle>';

    put '<PennedBy>';
/* Call the macro to replace "&" with "&amp;" to ensure the XML is valid */
    %fixAmpersand(Author);
    put Author;
    put '</PennedBy>';
put '</Book>';

/* at the end of the Category group, close the Category "node" in XML */
if Last.Category then do;
    put '</Category>';
end;

/* if reading the very last record, close the XML definition at the bottom */
if LastRecord then do;
    put '</Inventory>';
end;

run;
```