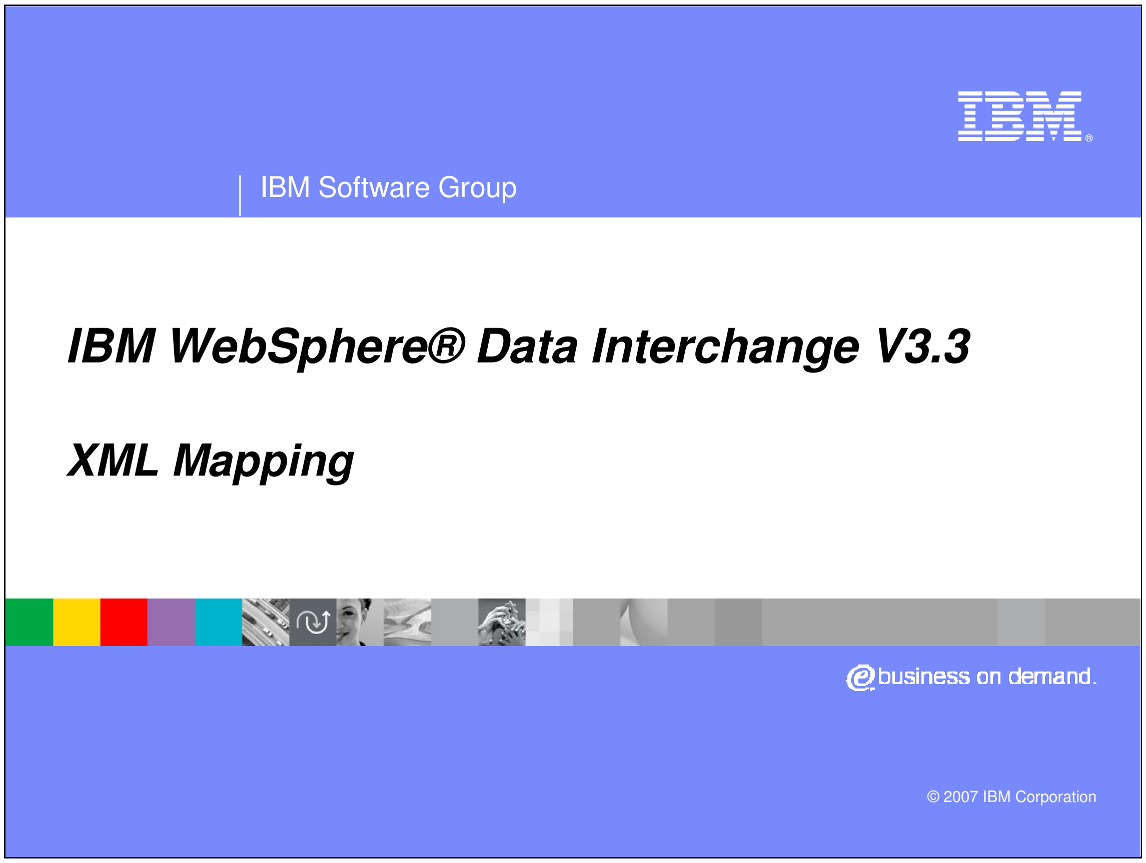


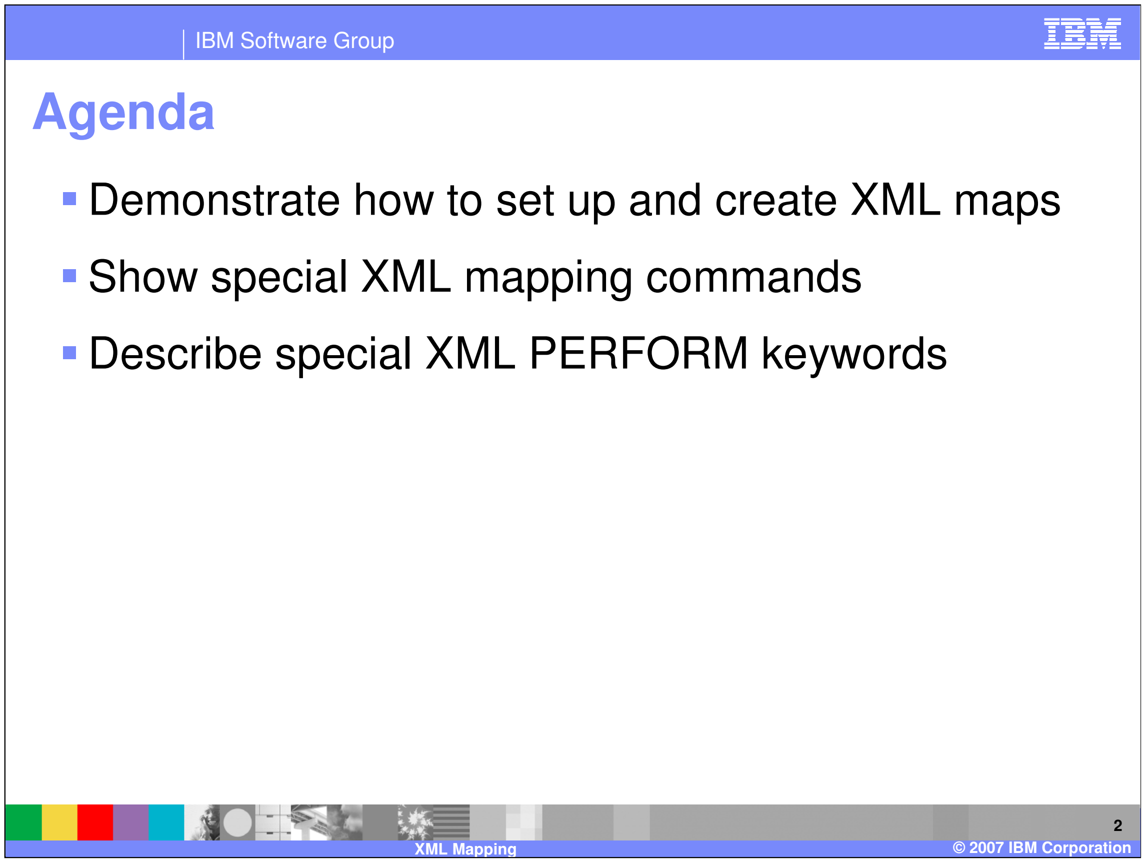
IBM Software Group

IBM WebSphere® Data Interchange V3.3

XML Mapping

business on demand.

This presentation will review XML mapping.

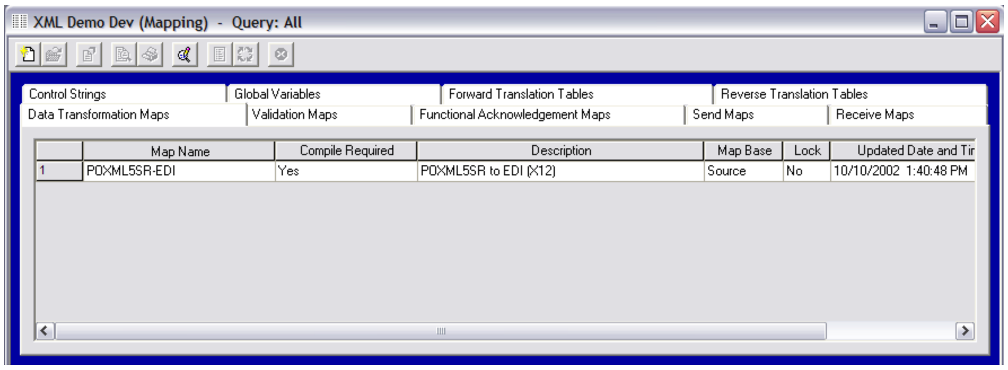


Agenda

- Demonstrate how to set up and create XML maps
- Show special XML mapping commands
- Describe special XML PERFORM keywords

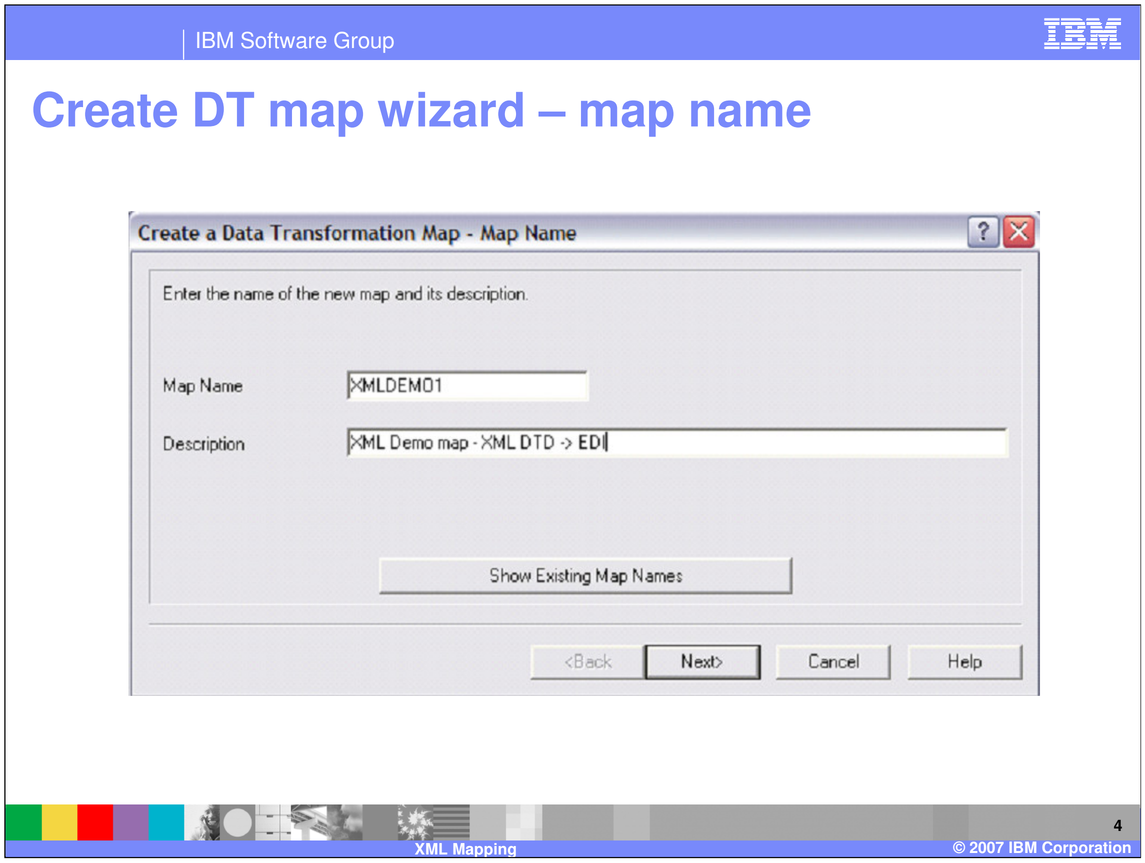
The presentation gives more detailed information about how to set up and create XML maps, special mapping commands that are used for XML data, and PERFORM command keywords that are specific to XML transformations.

Create XML map – DT map list window

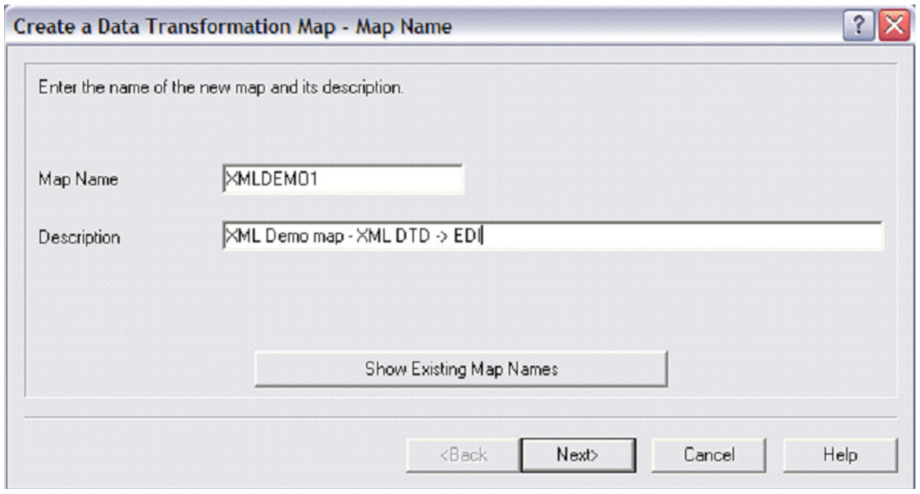

XML Demo Dev (Mapping) - Query: All
Control Strings
Global Variables
Forward Translation Tables
Reverse Translation Tables
Data Transformation Maps
Validation Maps
Functional Acknowledgement Maps
Send Maps
Receive Maps

	Map Name	Compile Required	Description	Map Base	Lock	Updated Date and Tir
1	POXML5SR-EDI	Yes	POXML5SR to EDI (X12)	Source	No	10/10/2002 1:40:48 PM

Creating an XML map is basically like creating any other Data Transformation (DT) map. You create the new map from the Data Transformation Maps tab of the Mapping functional area.



Create DT map wizard – map name


Create a Data Transformation Map - Map Name

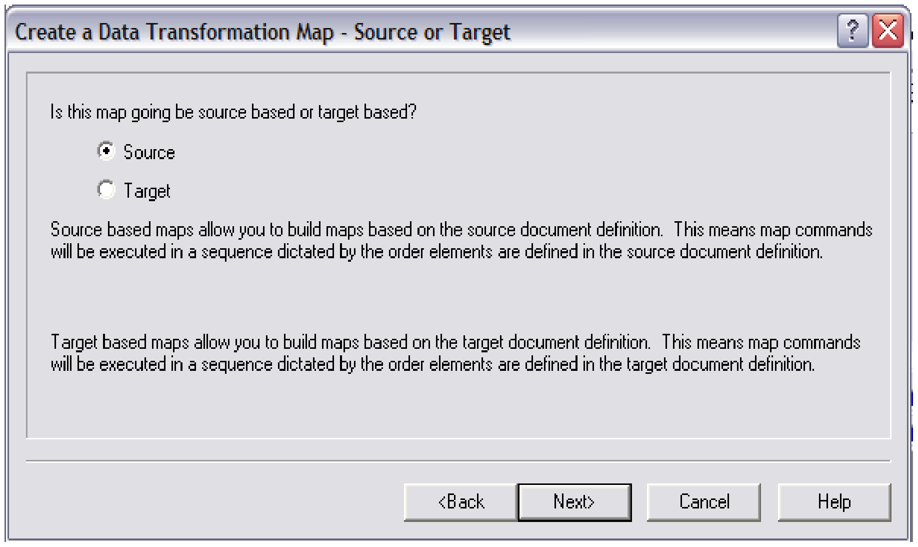
Enter the name of the new map and its description.

Map Name: XMLDEMO1

Description: XML Demo map - XML DTD -> EDI

Enter the Map name and optional description.

Create DT map wizard – source or target based



Create a Data Transformation Map - Source or Target

Is this map going be source based or target based?

(●) Source

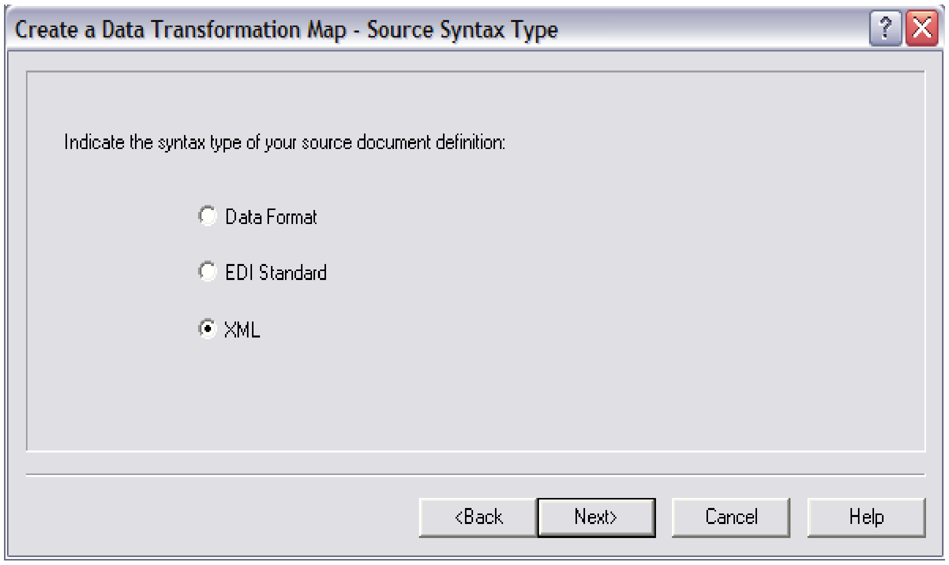
( ) Target

Source based maps allow you to build maps based on the source document definition. This means map commands will be executed in a sequence dictated by the order elements are defined in the source document definition.

Target based maps allow you to build maps based on the target document definition. This means map commands will be executed in a sequence dictated by the order elements are defined in the target document definition.

The map can be either source based or target based. Source based maps show the mapping commands relative to the source document definition, and target based maps show the commands relative to the target document definition.

Create DT map wizard – source syntax



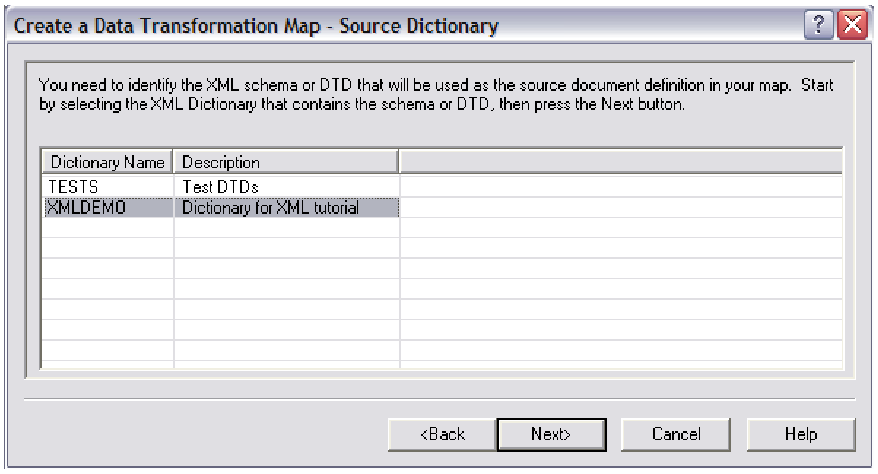
Create a Data Transformation Map - Source Syntax Type

Indicate the syntax type of your source document definition:

- ○ Data Format
- ○ EDI Standard
- [✓ XML]

This example is a XML to Electronic Data Interchange (EDI), so source syntax is XML.

Create DT map wizard – source dictionary

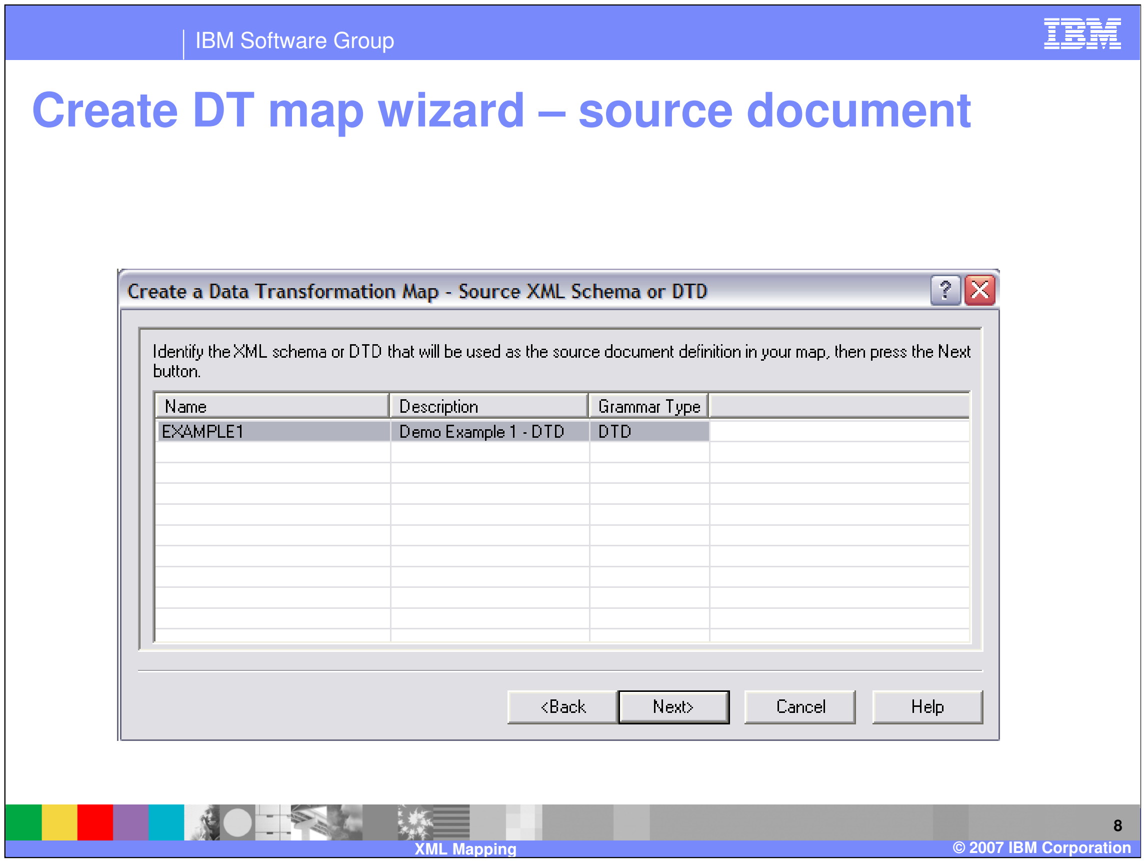


Create a Data Transformation Map - Source Dictionary

You need to identify the XML schema or DTD that will be used as the source document definition in your map. Start by selecting the XML Dictionary that contains the schema or DTD, then press the Next button.

Dictionary Name	Description
TESTS	Test DTDs
XMLDEMO	Dictionary for XML tutorial

Select the XML dictionary where the DTD or Schema is located.


IBM Software Group

Create DT map wizard – source document

Create a Data Transformation Map - Source XML Schema or DTD

Identify the XML schema or DTD that will be used as the source document definition in your map, then press the Next button.

Name	Description	Grammar Type
EXAMPLE1	Demo Example 1 - DTD	DTD

XML Mapping

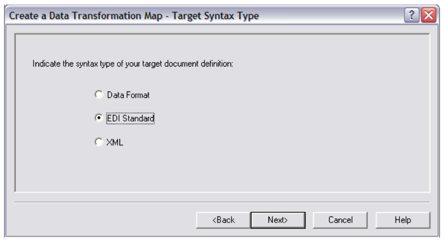
8

© 2007 IBM Corporation

Select the XML DTD or Schema definition that has been imported.

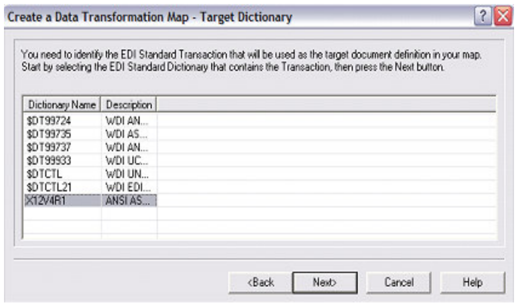
IBM Software Group

Create DT map wizard – target document



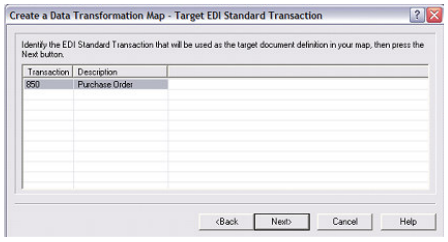
Indicate the syntax type of your target document definition:

Data Format [✓ EDI Standard] XML



You need to identify the EDI Standard Transaction that will be used as the target document definition in your map. Start by selecting the EDI Standard Dictionary that contains the Transaction, then press the Next button.

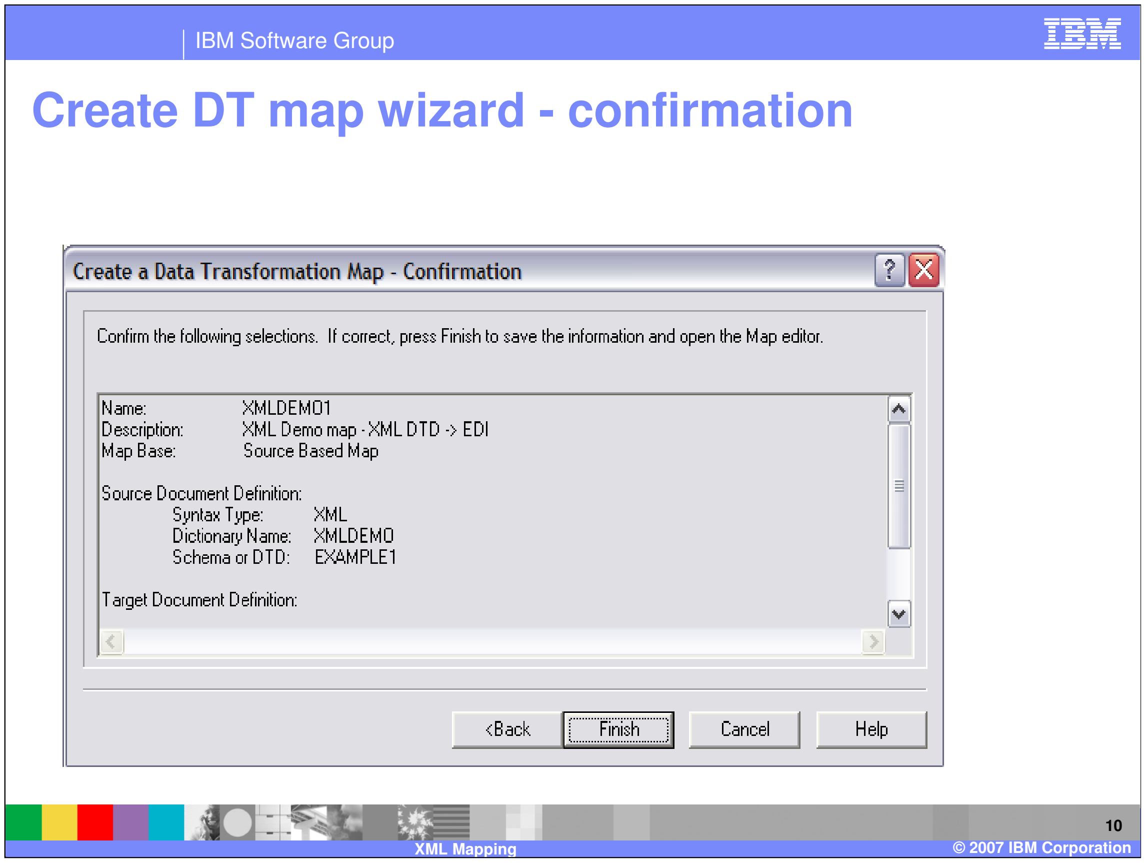
Dictionary Name	Description
$DT99724	WDI AN...
$DT99735	WDI AS...
$DT99737	WDI AN...
$DT99933	WDI UC...
$DTCTL	WDI UN...
$DTCTL21	WDI EDI...
X12V4R1	ANSI AS...


Create a Data Transformation Map - Target EDI Standard Transaction

Identify the EDI Standard Transaction that will be used as the target document definition in your map, then press the Next button.

Transaction	Description
850	Purchase Order

Continue on to choose the target document. In this example we will select X12 850.


IBM Software Group

Create DT map wizard - confirmation

Create a Data Transformation Map - Confirmation

Confirm the following selections. If correct, press Finish to save the information and open the Map editor.

```
Name:            XMLDEMO1
Description:     XML Demo map - XML DTD -> EDI
Map Base:        Source Based Map

Source Document Definition:
          Syntax Type:        XML
          Dictionary Name:    XMLDEMO
          Schema or DTD:      EXAMPLE1

Target Document Definition:
```

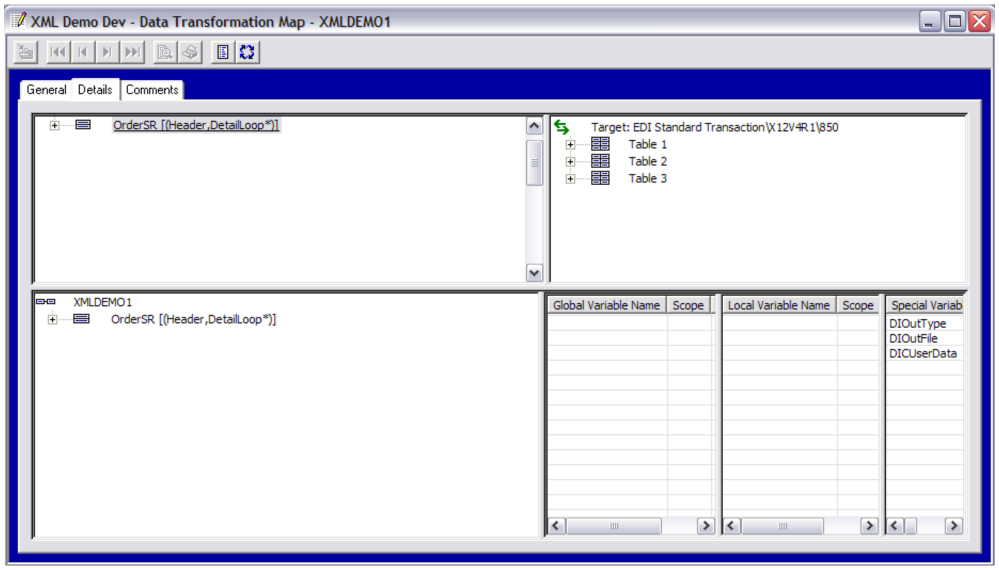
10

XML Mapping

© 2007 IBM Corporation

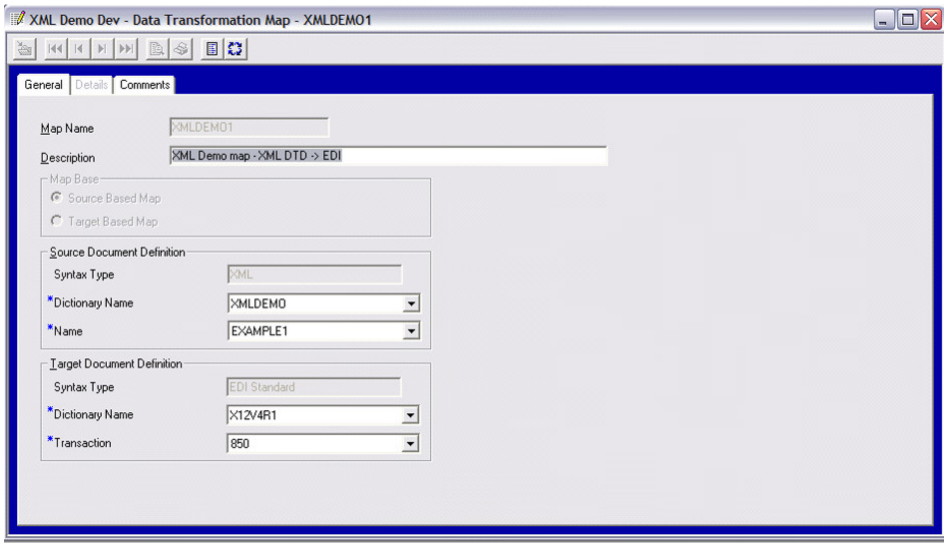
The map wizard asks you to confirm the information for the map. You can use the “Back” button to go back and change settings if needed.

DT map editor – Details tab


XML Demo Dev - Data Transformation Map - XMLDEMO1
General
Details
Comments
OrderSR [(Header,DetailLoop*)]
Target: EDI Standard Transaction\X12V4R1\850
Table 1
Table 2
Table 3
XMLDEMO1
OrderSR [(Header,DetailLoop*)]
Global Variable Name
Scope
Local Variable Name
Scope
Special Variab
DIOutType
DIOutFile
DICUserData

This is the Data Transformation mapping editor.

DT map editor – General tab



Map Name	XMLDEMO1
Description	XML Demo map - XML DTD -> EDI

Map Base
- (●) Source Based Map
- ( ) Target Based Map

Source Document Definition

Syntax Type	XML
*Dictionary Name	XMLDEMO
*Name	EXAMPLE1

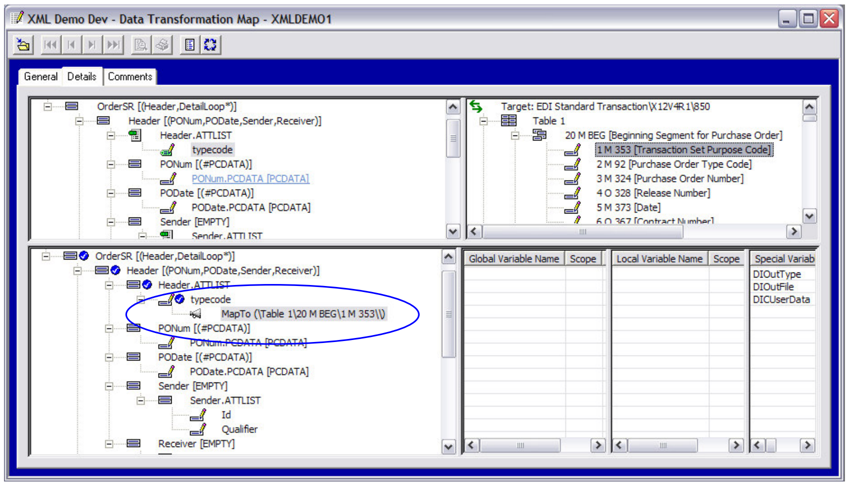
Target Document Definition

Syntax Type	EDI Standard
*Dictionary Name	X12V4R1
*Transaction	850

The General Tab contains the Source and Target document definitions.

IBM Software Group

Mapping – XML attributes



```
OrderSR [(Header,DetailLoop*)]
    Header [(PONum,PODate,Sender,Receiver)]
        Header.ATTLIST
            typecode
        PONum [(#PCDATA)]
            PONum.PCDATA [PCDATA]
        PODate [(#PCDATA)]
            PODate.PCDATA [PCDATA]
        Sender [EMPTY]
            Sender.ATTLIST
```

```
Target: EDI Standard Transaction\X12V4R1\850
    Table 1
        20 M BEG [Beginning Segment for Purchase Order]
            1 M 353 [Transaction Set Purpose Code]
            2 M 92 [Purchase Order Type Code]
            3 M 324 [Purchase Order Number]
            4 O 328 [Release Number]
            5 M 373 [Date]
            6 O 367 [Contract Number]
```

```
OrderSR [(Header,DetailLoop*)]
    Header [(PONum,PODate,Sender,Receiver)]
        Header.ATTLIST
            typecode
                MapTo (\Table 1\20 M BEG\1 M 353\\)
        PONum [(#PCDATA)]
            PONum.PCDATA [PCDATA]
        PODate [(#PCDATA)]
            PODate.PCDATA [PCDATA]
        Sender [EMPTY]
            Sender.ATTLIST
                Id
                Qualifier
        Receiver [EMPTY]
```

DIOutType
DIOutFile
DICUserData

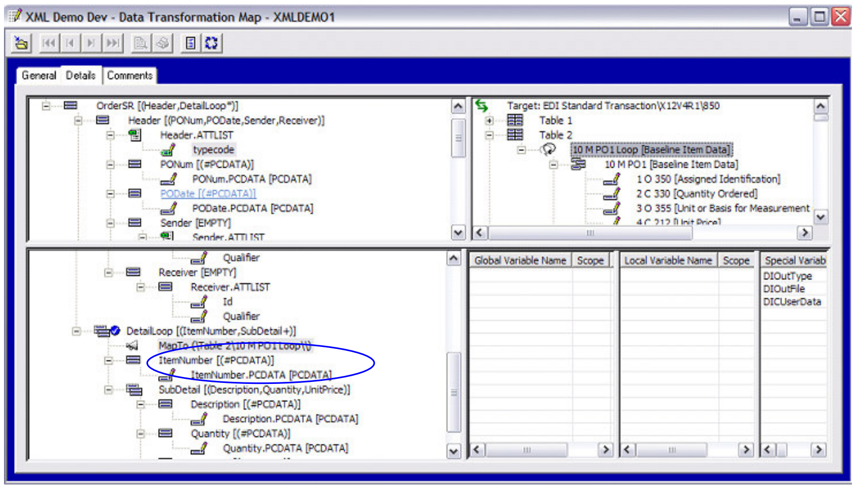
13

XML Mapping © 2007 IBM Corporation

You can use Drag-and-drop to create MapTo and MapFrom mapping commands. Note: Use the PCDATA node – NOT the element name.

Since this is a source based map, the mapping commands appear relative to the source XML document in the lower left pane.

Mapping – repeating elements



```
OrderSR [(Header,DetailLoop*)]
    Header [(PONum,PODate,Sender,Receiver)]
        Header.ATTLIST
            typecode
        PONum [(#PCDATA)]
            PONum.PCDATA [PCDATA]
        PODate [(#PCDATA)]
            PODate.PCDATA [PCDATA]
        Sender [EMPTY]
            Sender.ATTLIST
                Qualifier
        Receiver [EMPTY]
            Receiver.ATTLIST
                Id
                Qualifier
    DetailLoop [(ItemNumber,SubDetail+)]
        MapTo (\Table 2\10 M PO1 Loop\)
        ItemNumber [(#PCDATA)]
            ItemNumber.PCDATA [PCDATA]
        SubDetail [(Description,Quantity,UnitPrice)]
            Description [(#PCDATA)]
                Description.PCDATA [PCDATA]
            Quantity [(#PCDATA)]
                Quantity.PCDATA [PCDATA]
```

```
Target: EDI Standard Transaction\X12V4R1\850
    Table 1
    Table 2
        10 M PO1 Loop [Baseline Item Data]
            10 M PO1 [Baseline Item Data]
                1 O 350 [Assigned Identification]
                2 C 330 [Quantity Ordered]
                3 O 355 [Unit or Basis for Measurement
                4 C 212 [Unit Price]
```

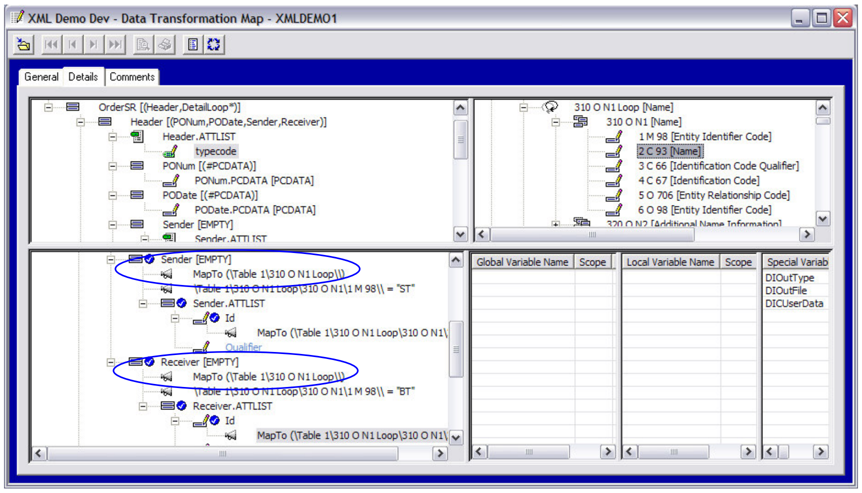
Global Variable Name	Scope

Local Variable Name	Scope

Special Variab
DIOutType
DIOutFile
DICUserData

Drag-and-drop for repeating elements creates MapTo or ForEach mapping commands and generates for example an output loop and record for each occurrence.

Mapping – repeating elements



XML Demo Dev - Data Transformation Map - XMLDEMO1

General | Details | Comments

```
OrderSR [(Header,DetailLoop*)]
    Header [(PONum,PODate,Sender,Receiver)]
        Header.ATTLIST
            typecode
        PONum [(#PCDATA)]
            PONum.PCDATA [PCDATA]
        PODate [(#PCDATA)]
            PODate.PCDATA [PCDATA]
        Sender [EMPTY]
            Sender.ATTLIST
```

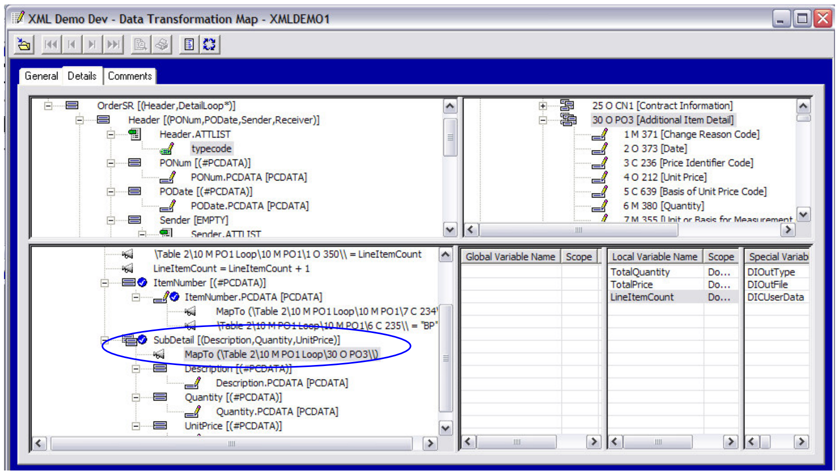
```
310 O N1 Loop [Name]
    310 O N1 [Name]
        1 M 98 [Entity Identifier Code]
        2 C 93 [Name]
        3 C 66 [Identification Code Qualifier]
        4 C 67 [Identification Code]
        5 O 706 [Entity Relationship Code]
        6 O 98 [Entity Identifier Code]
    320 O N2 [Additional Name Information]
```

```
Sender [EMPTY]
    MapTo (\Table 1\310 O N1 Loop\\)
    ~~\Table 1\310 O N1 Loop\310 O N1\1 M 98\\ = "ST"~~
    Sender.ATTLIST
        Id
            MapTo (\Table 1\310 O N1 Loop\310 O N1\
        Qualifier
Receiver [EMPTY]
    MapTo (\Table 1\310 O N1 Loop\\)
    ~~\Table 1\310 O N1 Loop\310 O N1\1 M 98\\ = "BT"~~
    Receiver.ATTLIST
        Id
            MapTo (\Table 1\310 O N1 Loop\310 O N1\
```

Global Variable Name	Scope	Local Variable Name	Scope	Special Variab
				DIOutType
				DIOutFile
				DICUserData

The source element does not necessarily need to repeat in order to get repeating items in the target. This example creates an occurrence of the N1 loop for Sender, and another for Receiver.

Mapping – nested repeating elements



```
OrderSR [(Header,DetailLoop*)]
    Header [(PONum,PODate,Sender,Receiver)]
        Header.ATTLIST
            typecode
        PONum [(#PCDATA)]
            PONum.PCDATA [PCDATA]
        PODate [(#PCDATA)]
            PODate.PCDATA [PCDATA]
        Sender [EMPTY]
            Sender.ATTLIST
```

```
25 O CN1 [Contract Information]
30 O PO3 [Additional Item Detail]
    1 M 371 [Change Reason Code]
    2 O 373 [Date]
    3 C 236 [Price Identifier Code]
    4 O 212 [Unit Price]
    5 C 639 [Basis of Unit Price Code]
    6 M 380 [Quantity]
    7 M 355 [Unit or Basis for Measurement
```

```
    \Table 2\10 M PO1 Loop\10 M PO1\1 O 350\\ = LineItemCount
    LineItemCount = LineItemCount + 1
ItemNumber [(#PCDATA)]
    ItemNumber.PCDATA [PCDATA]
        MapTo (\Table 2\10 M PO1 Loop\10 M PO1\7 C 234'
        \Table 2\10 M PO1 Loop\10 M PO1\6 C 235\\ = "BP"
SubDetail [(Description,Quantity,UnitPrice)]
    MapTo (\Table 2\10 M PO1 Loop\30 O PO3\\)
    Description [(#PCDATA)]
        Description.PCDATA [PCDATA]
    Quantity [(#PCDATA)]
        Quantity.PCDATA [PCDATA]
    UnitPrice [(#PCDATA)]
```

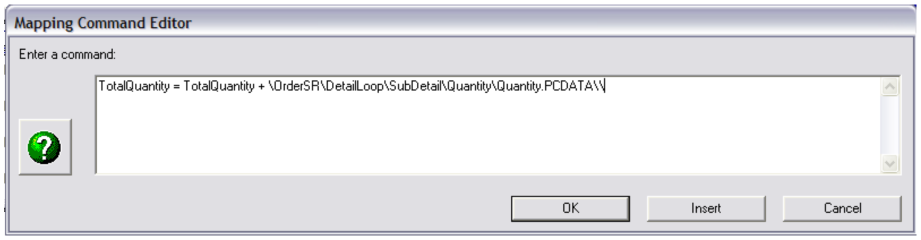
Global Variable Name	Scope

Local Variable Name	Scope
TotalQuantity	Do...
TotalPrice	Do...
LineItemCount	Do...

Special Variab
DIOutType
DIOutFile
DICUserData

You can do multi-occurrence mapping on nested elements. This example creates a PO3 loop in PO1 loop for each SubDetail in DetailLoop.

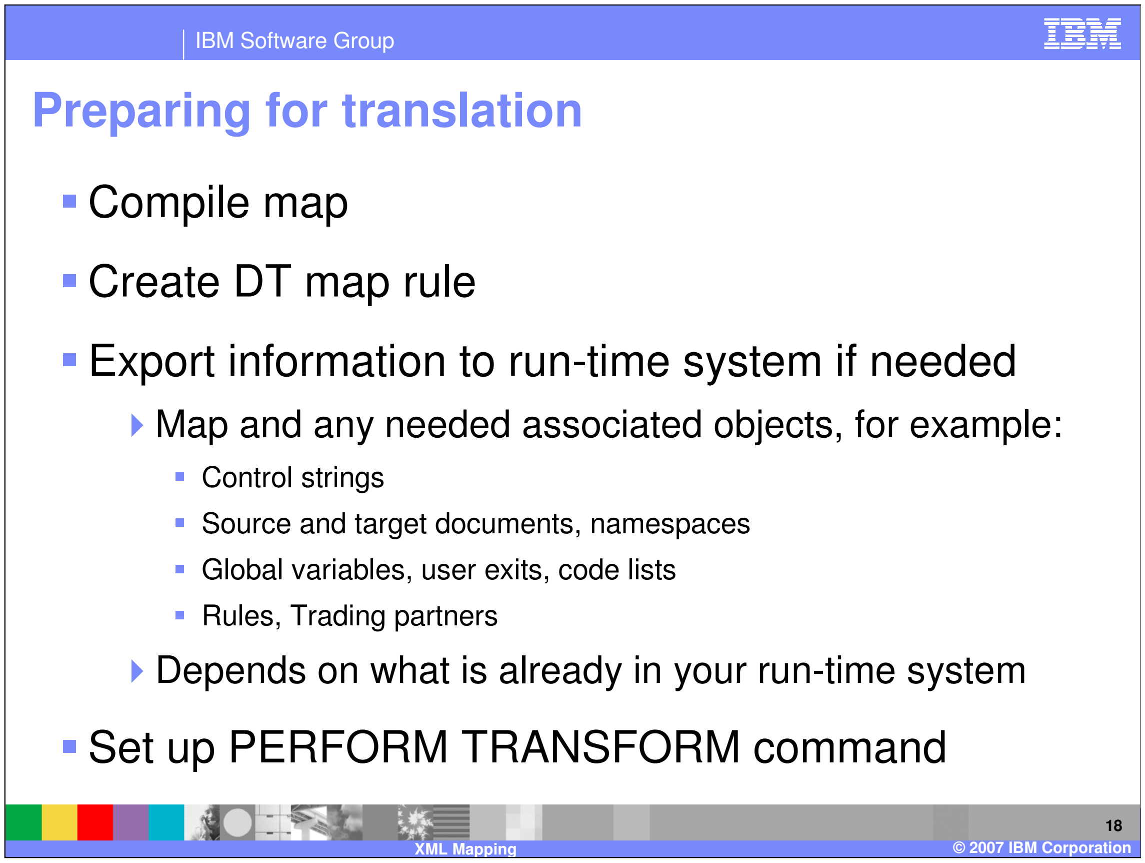
Mapping – using XML values in expressions



```
TotalQuantity = TotalQuantity + \OrderSR\DetailLoop\SubDetail\Quantity\Quantity.PCDATA\\
```

You can use element values and attribute values in expressions, just like EDI data elements and Data Format fields. For example, you can create an assignment command in the map then Drag and drop elements into command to create paths.

Again, use the PCDATA node – NOT the element name.



Preparing for translation

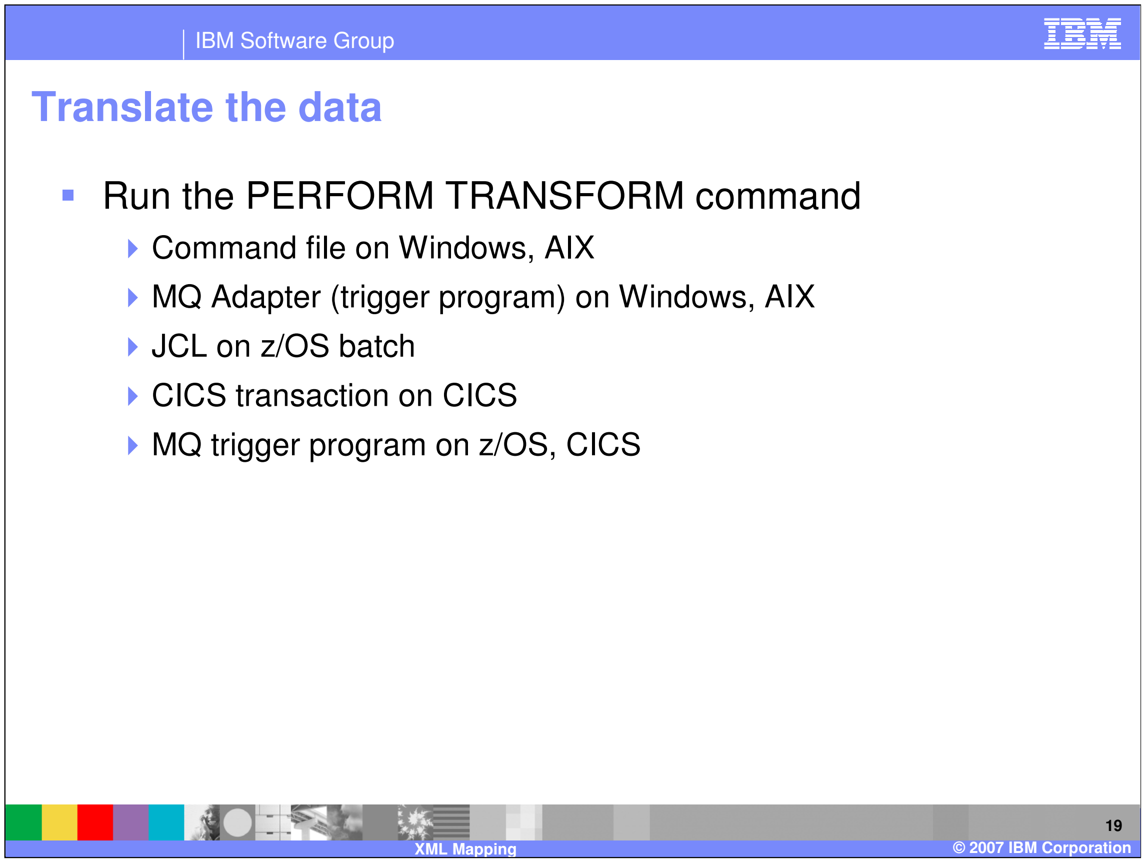
- Compile map
- Create DT map rule
- Export information to run-time system if needed
  - Map and any needed associated objects, for example:
    - Control strings
    - Source and target documents, namespaces
    - Global variables, user exits, code lists
    - Rules, Trading partners
  - Depends on what is already in your run-time system
- Set up PERFORM TRANSFORM command

Once you have created and saved the map, you need to compile it using the WDI Client. This converts the information in the map to an internal byte code called a *control string*, which significantly improves run-time performance.

You also need to create a Data Transformation (DT) map rule. This tells WDI *when* this map should be used. You may want to set up different rules so you can do different transformations based on which trading partner the document came from.

You either create a map directly in the database that is used by WDI Server at run-time, or you may create it in a local database. If you created it in a local database, you will need to export it from the local database and import it into the database used by WDI Server. Other associated objects such as control strings, source and target document definitions, etc. may also need to be exported and imported into the server database. Many of these objects can be shared by multiple maps, so if they have not changed then you can reuse the associated objects that are already in the server database.

Finally, you set up your PERFORM TRANSFORM command.



Translate the data

- Run the PERFORM TRANSFORM command
  - Command file on Windows, AIX
  - MQ Adapter (trigger program) on Windows, AIX
  - JCL on z/OS batch
  - CICS transaction on CICS
  - MQ trigger program on z/OS, CICS

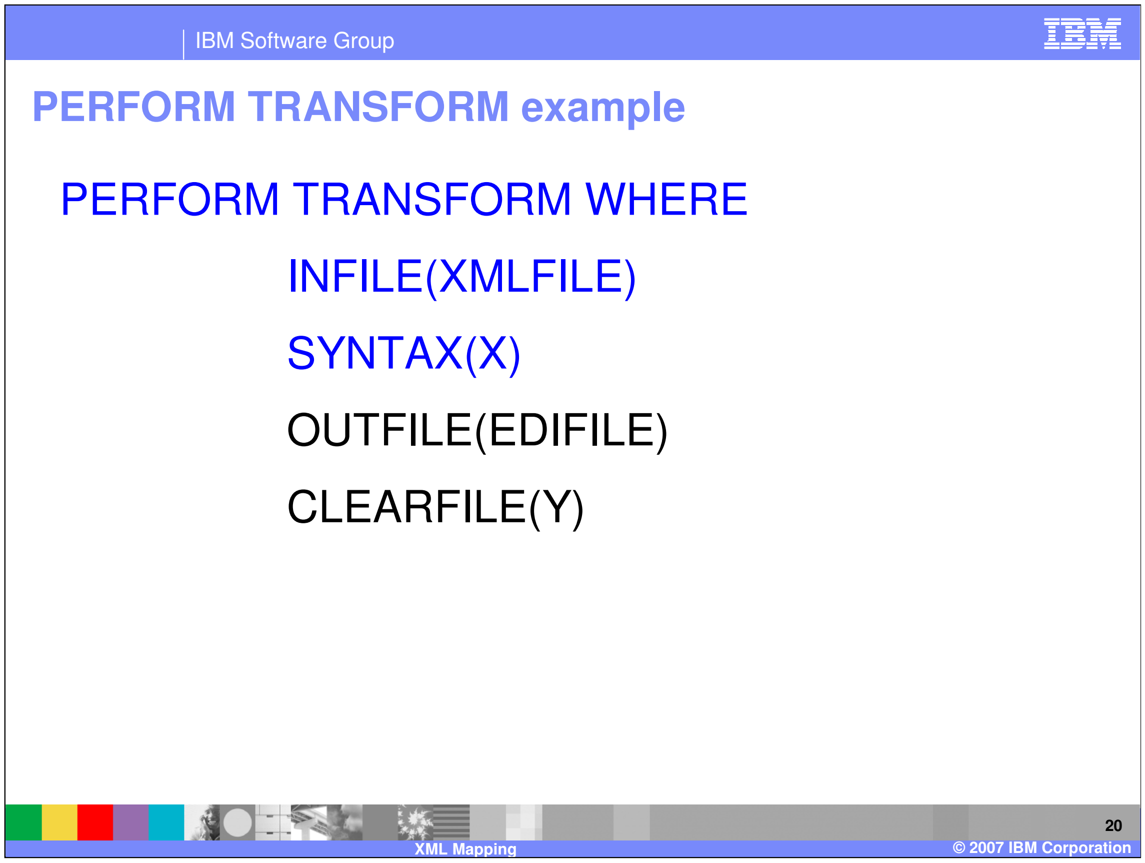
To translate the data, you run the WDI Server and pass it the PERFORM TRANSFORM command. WDI server can be started in different ways, depending on the platform.

-On Windows and AIX, the server is often started from the command line using a command file as input, or the server may be triggered from an MQ queue using the WDI Adapter.

-For z/OS batch, the server is typically run using JCL

-In CICS, it is usually started by a CICS transaction

-MQ trigger programs can also be used to start the server on both z/OS and CICS



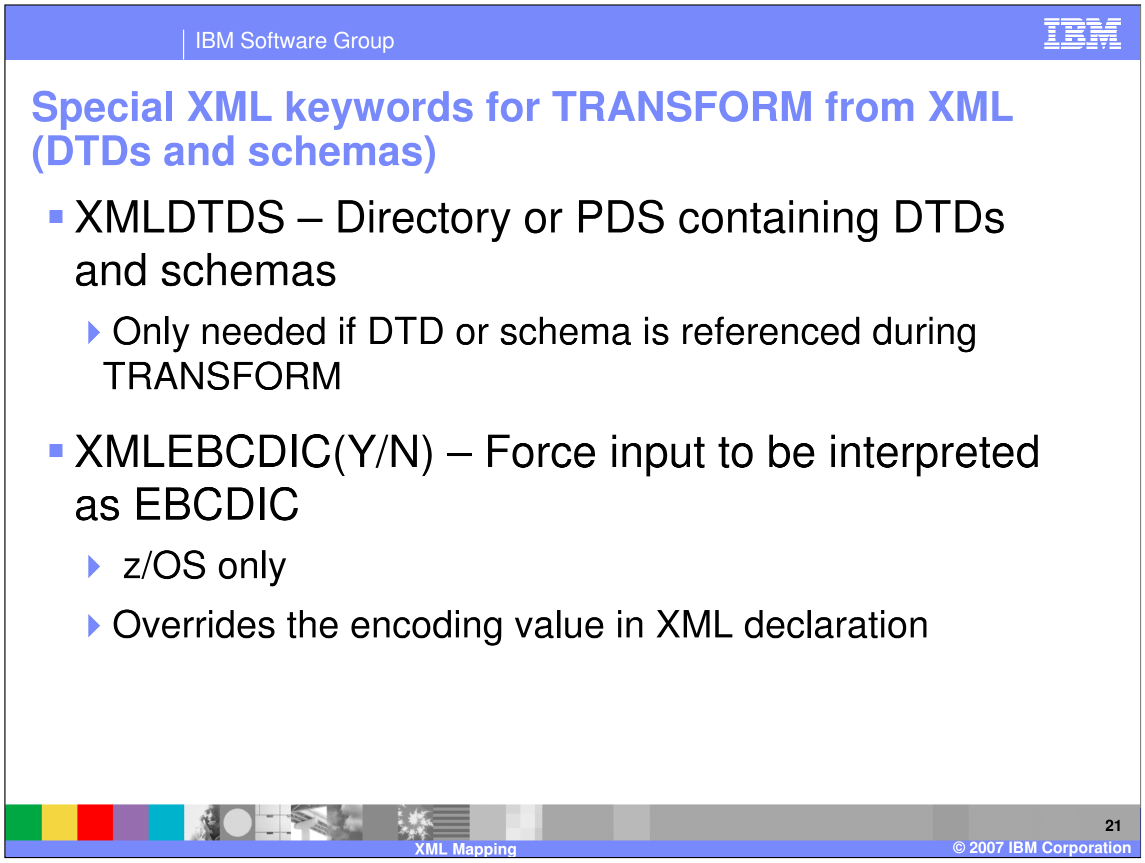
PERFORM TRANSFORM example

```
PERFORM TRANSFORM WHERE
          INFILE(XMLFILE)
          SYNTAX(X)
          OUTFILE(EDIFILE)
          CLEARFILE(Y)
```

The PERFORM TRANSFORM is passed to the WDI Server, and tells the server that it is to process the data using a Data Transformation map.

This is a sample PERFORM TRANSFORM command. The SYNTAX and INFILE keywords are required. These specify the syntax for the XML source data, and the input file name, which is a DD name or logical file name. OUTFILE and CLEARFILE keywords are also commonly used. OUTFILE is the DD name or logical file name for the output file, and CLEARFILE indicates that the data in the output file should be replaced instead of being appended to.

The "WebSphere Data Interchange Utility Commands and File Formats Reference" describes all keywords that are supported.



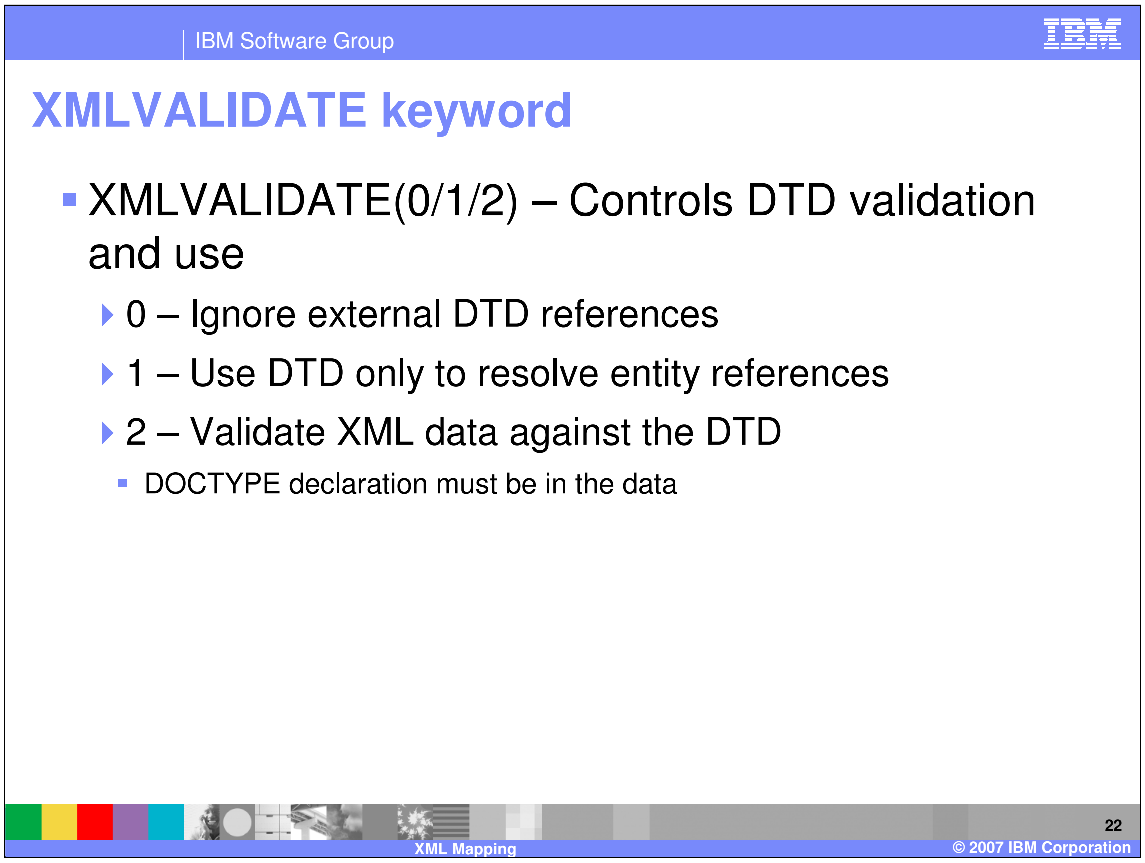
Special XML keywords for TRANSFORM from XML (DTDs and schemas)

- XMLDTDS – Directory or PDS containing DTDs and schemas
  - Only needed if DTD or schema is referenced during TRANSFORM
- XMLEBCDIC(Y/N) – Force input to be interpreted as EBCDIC
  - z/OS only
  - Overrides the encoding value in XML declaration

There are some PERFORM TRANSFORM keywords that apply specifically to processing XML source documents.

XMLDTDS is used to tell WDI which directory or partitioned data set contains the XML schemas or DTDs. These copies of the schemas and DTDs are only used if they are referenced during the TRANSFORM. For example, they may be read in order to validate that the XML document conforms to the schema or DTD. This type of validation uses the file, instead of the schema or DTD that was imported into the database.

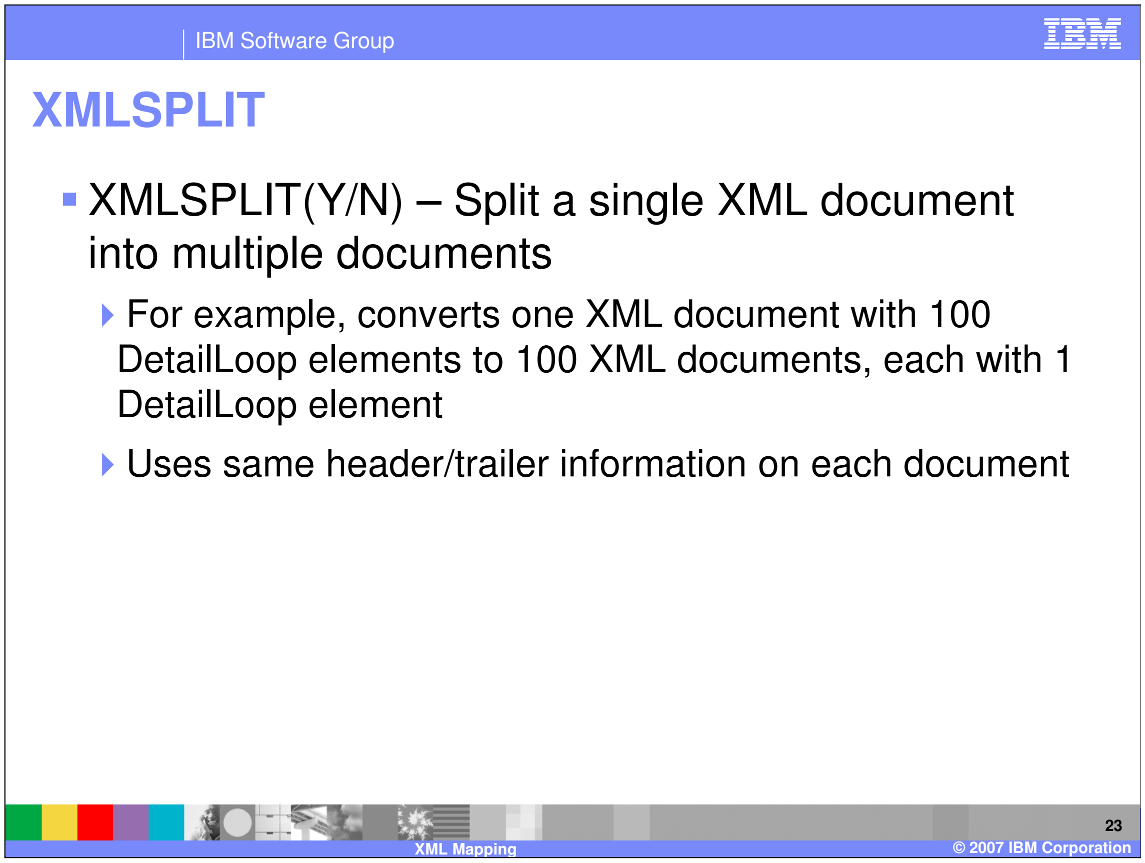
XMLEBCDIC is only used on z/OS, including CICS. This tells WDI that it should interpret the XML document as EBCDIC, regardless of the encoding name in the XML declaration. This can be useful if the encoding value in the XML data is incorrect because it was created on an ASCII platform, then converted to EBCDIC when it was transferred to the z/OS system.



XMLVALIDATE keyword

- XMLVALIDATE(0/1/2) – Controls DTD validation and use
  - 0 – Ignore external DTD references
  - 1 – Use DTD only to resolve entity references
  - 2 – Validate XML data against the DTD
    - DOCTYPE declaration must be in the data

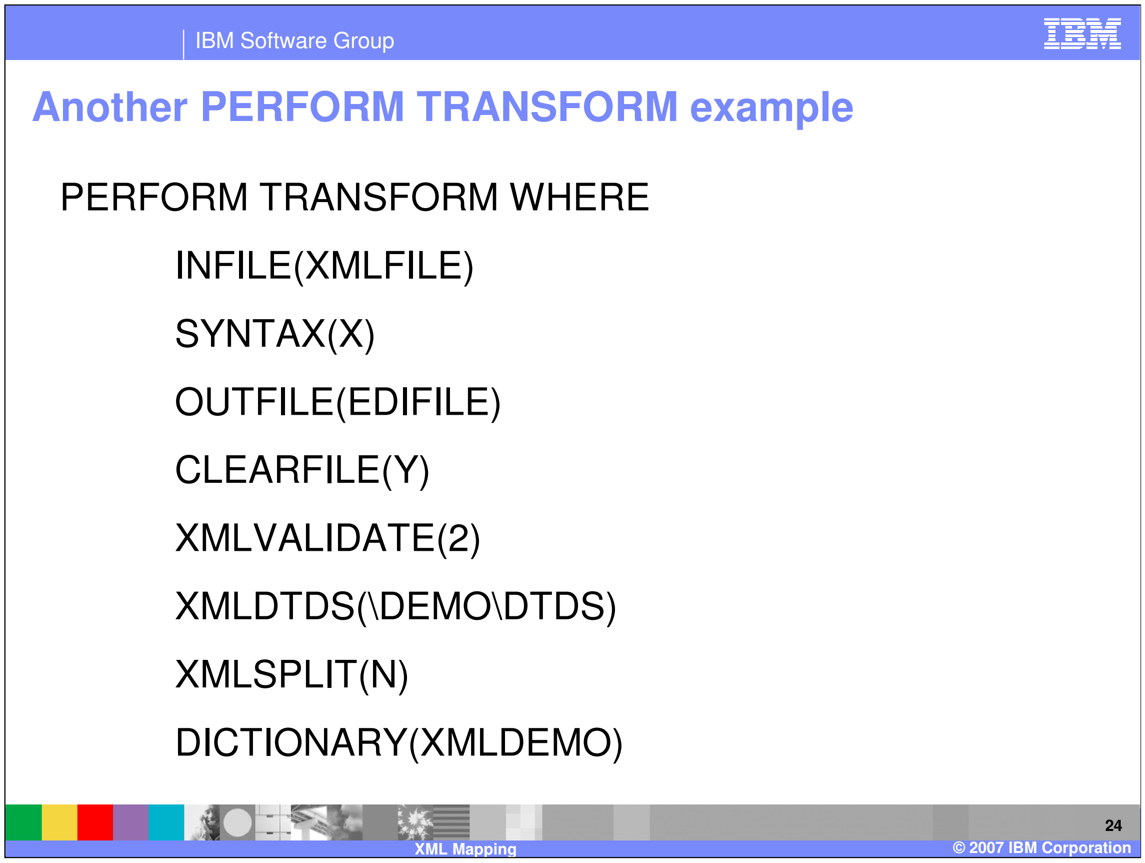
XMLVALIDATE controls whether the DTD is used or not, and whether the DTD is used for validation or only to resolve entity references.



XMLSPLIT

- XMLSPLIT(Y/N) – Split a single XML document into multiple documents
  - For example, converts one XML document with 100 DetailLoop elements to 100 XML documents, each with 1 DetailLoop element
  - Uses same header/trailer information on each document

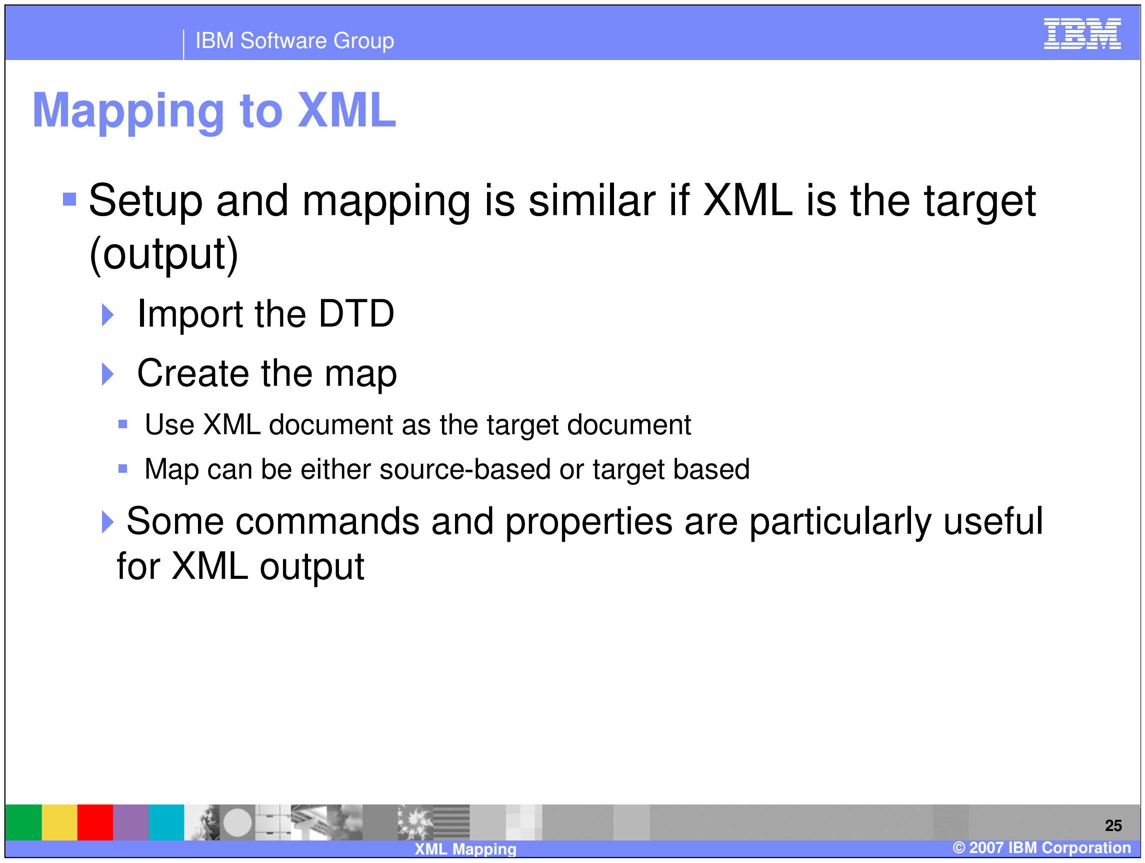
The XMLSPLIT keyword is used to break a single XML document into multiple documents. Each document uses the header and trailer information from the original source document, and has one occurrence of a given repeating element. There is more detail on this in a separate presentation.



Another PERFORM TRANSFORM example

```
PERFORM TRANSFORM WHERE
        INFILE(XMLFILE)
        SYNTAX(X)
        OUTFILE(EDIFILE)
        CLEARFILE(Y)
        XMLVALIDATE(2)
        XMLDTDS(\DEMO\DTDS)
        XMLSPLIT(N)
        DICTIONARY(XMLDEMO)
```

Here is another example of the PERFORM TRANSFORM command, with some of the XML specific keywords. The DICTIONARY keyword in this example can be used for other syntaxes. In this case, when WDI Server tries to determine the schema or DTD based on the root element, it will only look in the XMLDEMO dictionary.


IBM Software Group

Mapping to XML

- Setup and mapping is similar if XML is the target (output)
  - Import the DTD
  - Create the map
    - Use XML document as the target document
    - Map can be either source-based or target based
  - Some commands and properties are particularly useful for XML output

25

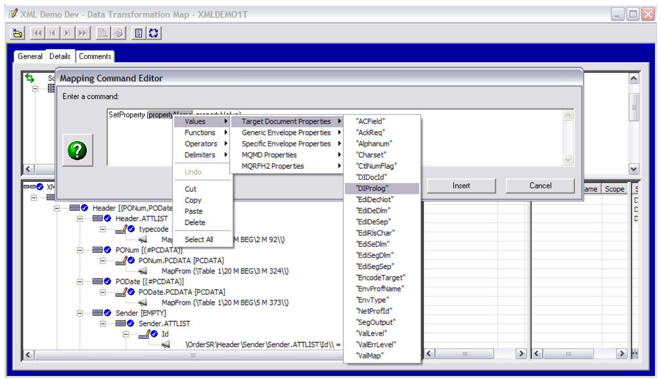
XML Mapping

© 2007 IBM Corporation

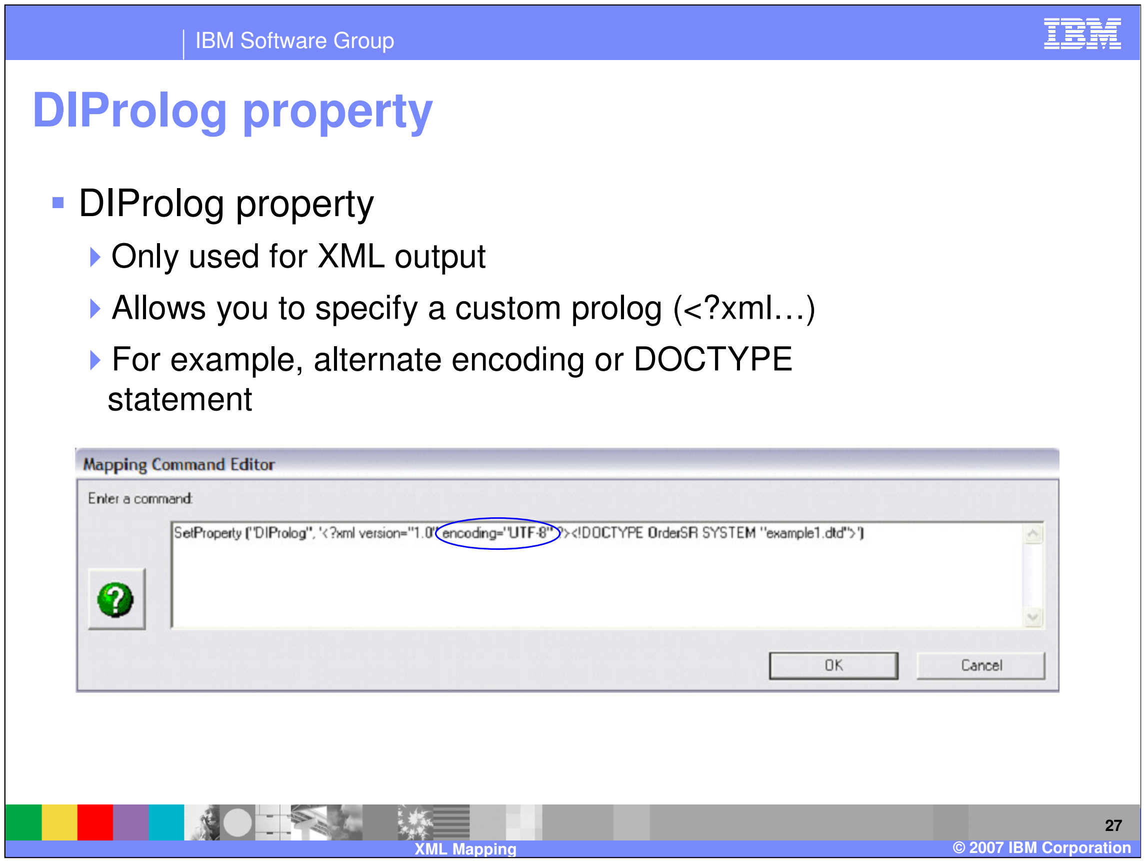
If you are mapping to an XML target document, the process is similar. However, there are some special mapping commands and properties that you may want to use when you generate XML output.

Special XML mapping properties

- Some properties only apply to, or are more useful for XML data
- Right-click on propertyName for list of available properties



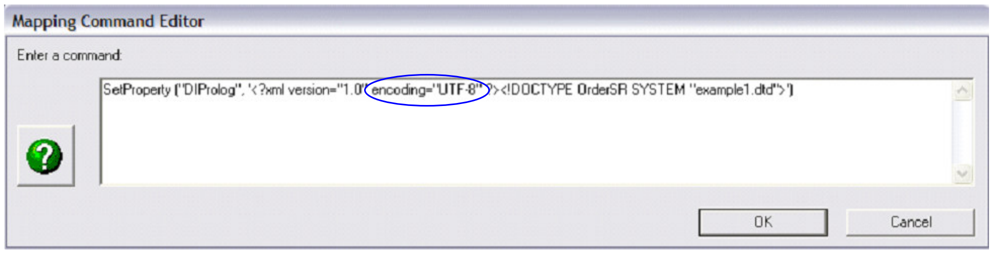
Use the SetProperty mapping command to set the property. If you right-click on the PropertyName, WDI Client will display a list of property names you can select from.



IBM Software Group

DIProlog property

- DIProlog property
  - Only used for XML output
  - Allows you to specify a custom prolog (<?xml…)
  - For example, alternate encoding or DOCTYPE statement



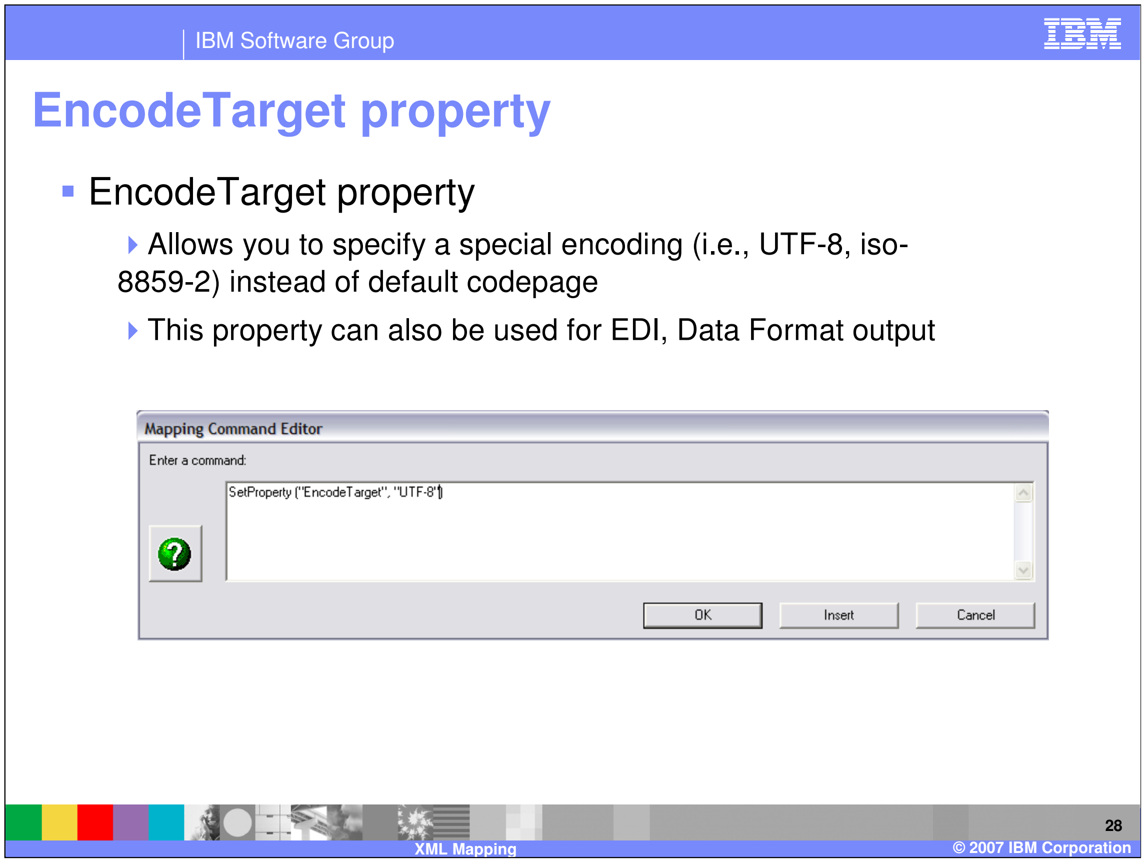
```
SetProperty ('DIProlog', '<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE OrderSR SYSTEM "example1.dtd">')
```

27

XML Mapping

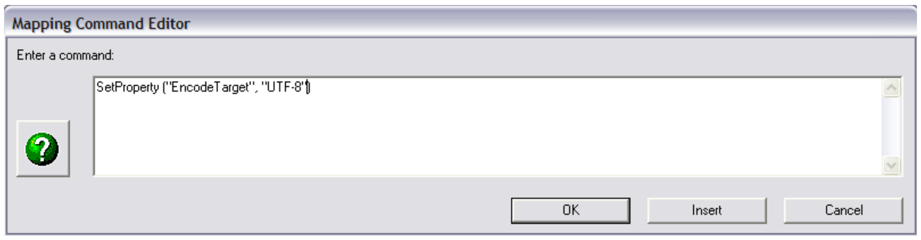
© 2007 IBM Corporation

The DIProlog property allows you to specify a custom prolog. This can be useful if you need to specify a particular encoding or DOCTYPE statement.



EncodeTarget property

- EncodeTarget property
  - Allows you to specify a special encoding (i.e., UTF-8, iso-8859-2) instead of default codepage
  - This property can also be used for EDI, Data Format output

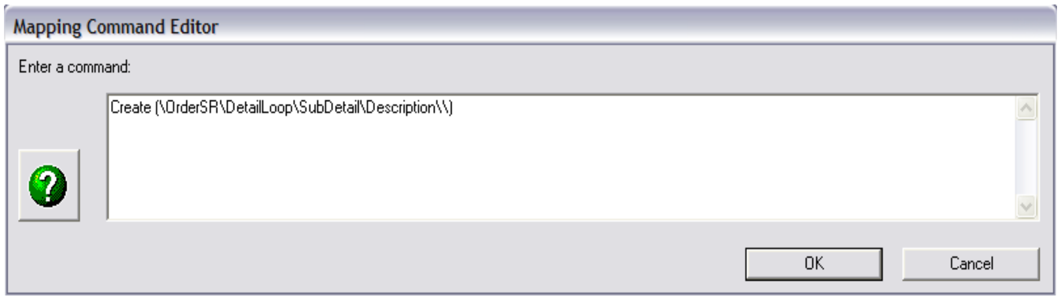


```
SetProperty ("EncodeTarget", "UTF-8")
```

EncodeTarget allows you to specify a special encoding such as UTF-8 or UTF-16 for the output data. This can also be used for other types of output such as EDI and record-oriented Data Format data. However, it is more commonly used for XML, since the use of Unicode and international character sets seems to be more prevalent with XML data than with EDI.

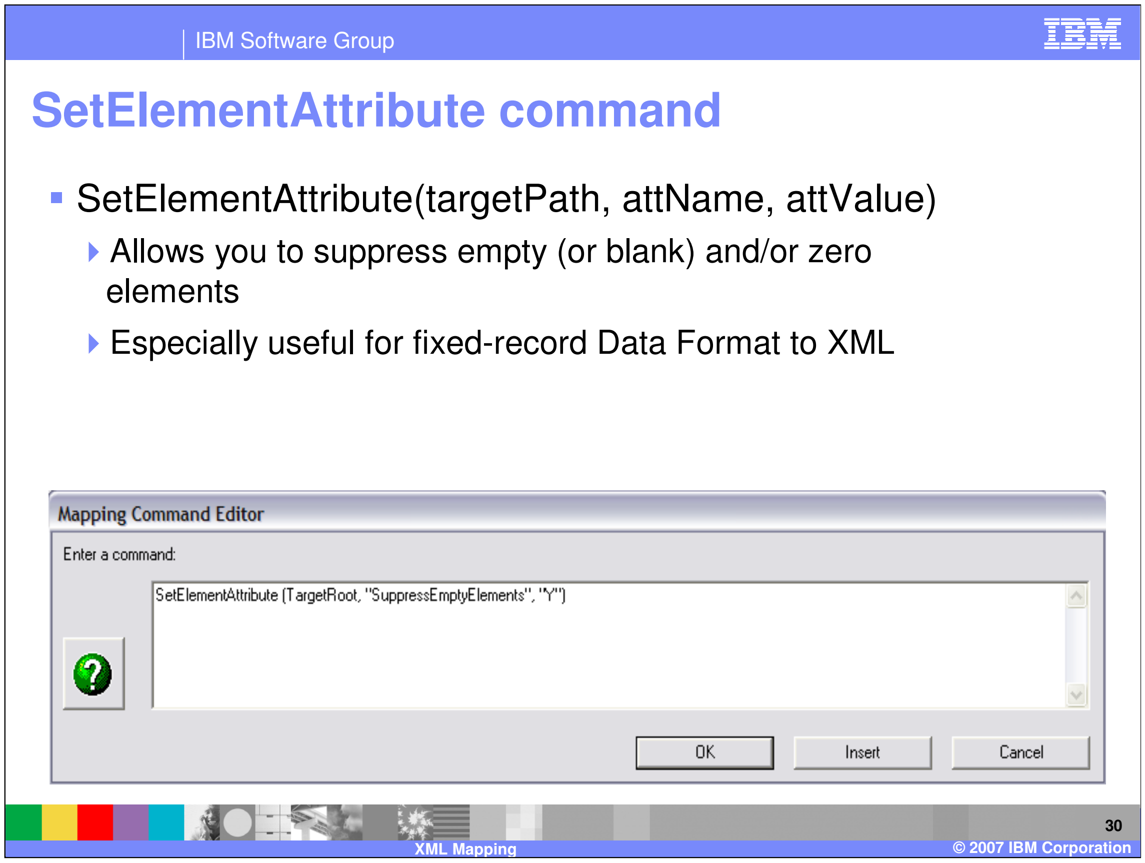
Create command

- Create(targetPath)
  - Allows you to create an empty element with no attributes



```
Create (\OrderSR\DetailLoop\SubDetail\Description\\)
```

The Create command allows you to generate an empty element with no attributes. This may be needed in cases where a schema or DTD indicates that an element is mandatory, but allows it to be empty. (Of course, defining an element this way is probably not considered a "best practice", but it does happen sometimes.)



IBM Software Group

SetElementAttribute command

- SetElementAttribute(targetPath, attName, attValue)
  - Allows you to suppress empty (or blank) and/or zero elements
  - Especially useful for fixed-record Data Format to XML

Mapping Command Editor

Enter a command:

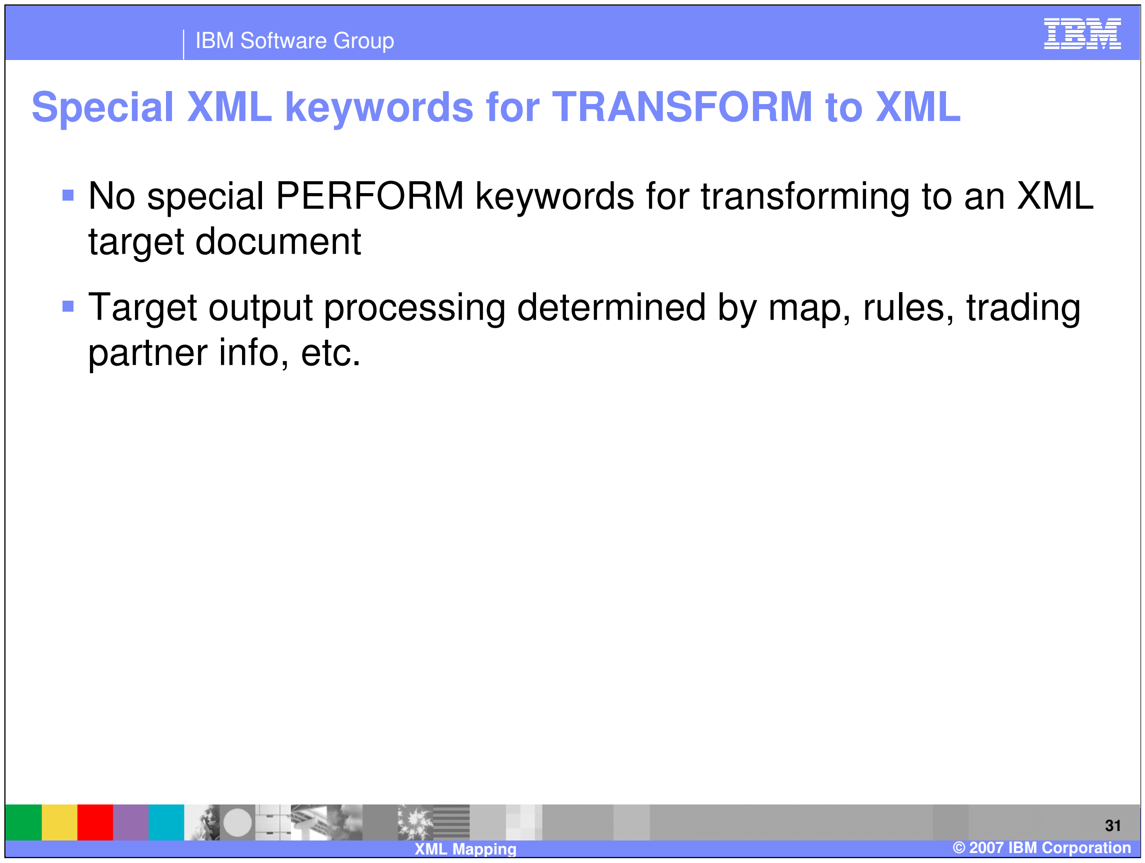
```
SetElementAttribute (TargetRoot, "SuppressEmptyElements", "Y")
```

30

XML Mapping

© 2007 IBM Corporation

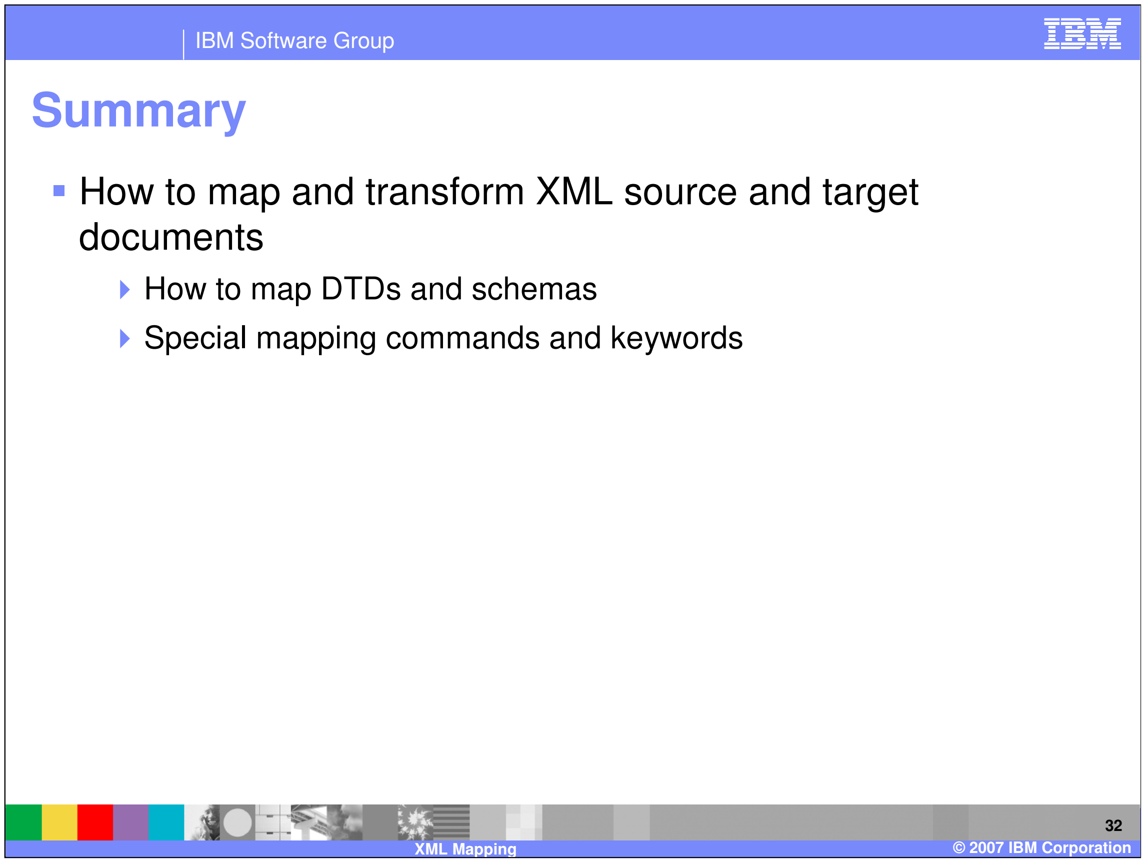
SetElementAttribute allows you to suppress empty, blank, and/or zero elements from the output. This mapping command can also be used for other syntaxes besides XML, but is particularly useful when mapping fixed-record Data Format definitions to XML. In this case, many of the fields in the fixed-record Data Format input may be blank. SetElementAttribute can be used to suppress these elements, instead of generating XML elements that contain blank values.



Special XML keywords for TRANSFORM to XML

- No special PERFORM keywords for transforming to an XML target document
- Target output processing determined by map, rules, trading partner info, etc.

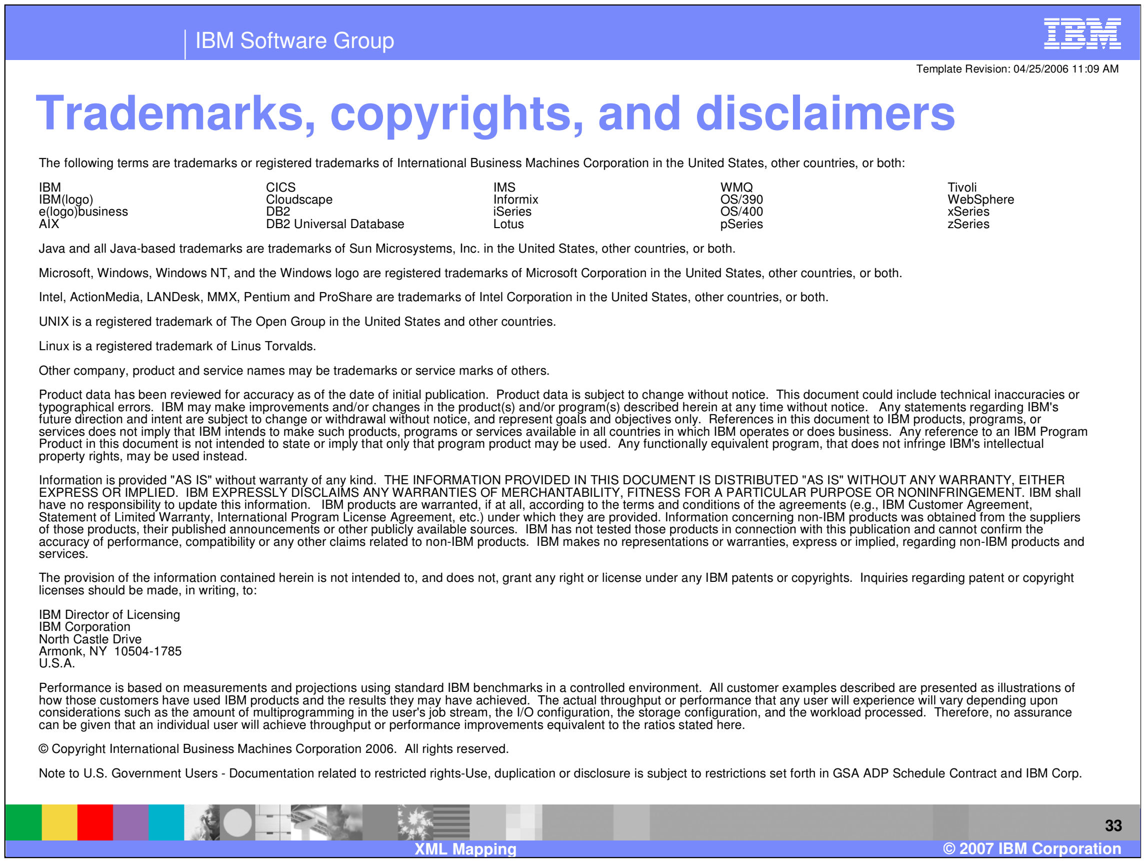
There are no special keywords for using the PERFORM TRANSFORM server command to generate XML output. The target output processing is determined by the map, rules, trading partner information, etc.



Summary

- How to map and transform XML source and target documents
  - How to map DTDs and schemas
  - Special mapping commands and keywords

This presentation showed how to create a map that uses an XML DTD or schema to define the source or target document.   It also described some mapping commands and PERFORM TRANSFORM keywords that can be particularly useful for processing XML data. There are also some other XML-specific keywords that apply to XML schemas and to namespaces. These keywords are covered in other presentations.



Trademarks, copyrights, and disclaimers