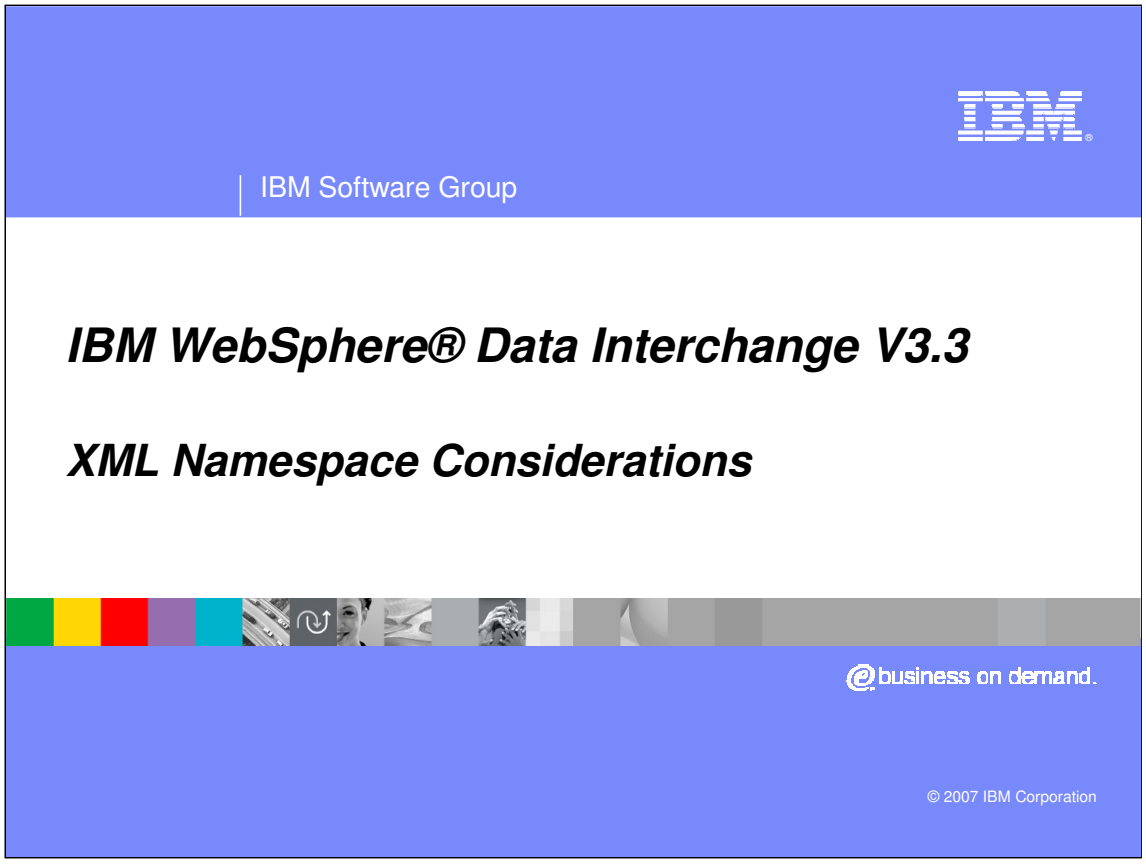


This presentation will review XML Namespace considerations.

Agenda

- Review XML namespace concepts and terminology
- Show special considerations when using XML namespaces with WDI

The presentation describes some special considerations for using XML namespaces with WebSphere Data Interchange (WDI). We will start with a review of some XML namespace concepts and terminology. Then talk about how these apply to WDI, including differences in setup, mapping, and transformation.

XML namespaces

- Help to resolve name conflicts when multiple schemas are used to define a document
- Defined by xmlns:prefix attribute
  - Example: xmlns:po="http://example.com/POExample"
  - Defines "po" as prefix for elements and attributes in this namespace. i.e., <po:Address>
- "Namespace aware" applications process elements and attributes based on the *namespace*, not the *prefix*
  - i.e., Internally <http://example.com/POExample:Address>

Sometimes multiple schemas are used to define a document. Namespaces are used to resolve name conflicts between these schemas. For example, one schema may use the element name "Address" to indicate a postal address, while another schema uses it as an e-mail address. The namespace can be used to specify which schema a particular instance of the element belongs to.

A prefix is associated with the namespace by using the **xmlns:prefix** attribute, like in this example**.** In this case, the part starting with **http:** is the namespace, and **po** is the prefix. So elements that belong to that namespace would have a prefix of "**po:**".

A "namespace aware" application processes the elements based on the namespace, not the prefix. So internally, the application would treat the element name in this example as: <http://example.com/POExample:Address>. When you use the XMLNS(Y) PERFORM keyword option, WebSphere Data Interchange (WDI) acts as a "namespace aware" application. We will discuss that option more later.

Defining namespace prefixes in the schema

- Defines a shorthand notation for elements that belong to a Particular namespace
  - The prefix only applies to this document

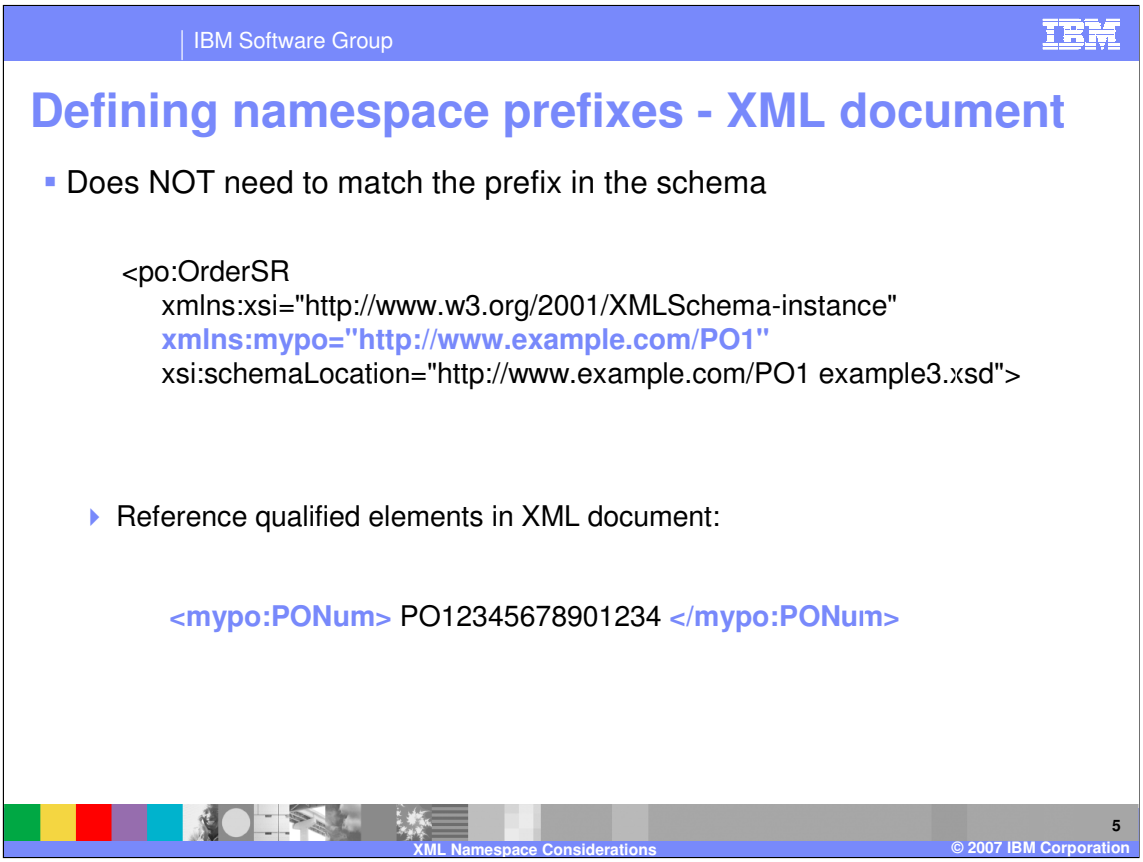
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:po="http://www.example.com/PO1"
            xmlns:common="http://www.example.com/common"
            targetNamespace="http://www.example.com/PO1"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
```

  - Reference qualified elements in schema:

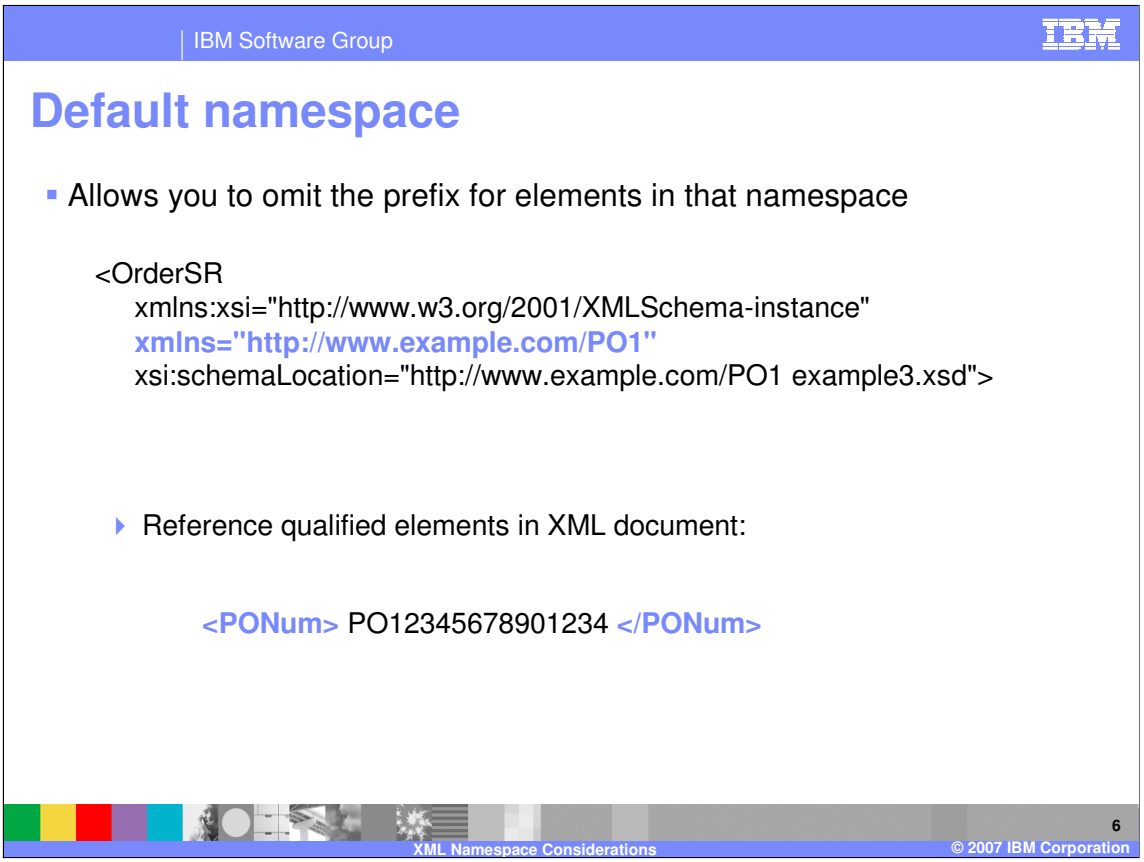
```
po:HeaderType
common:TradingPartner
```


The prefix defines a shorthand notation for elements that belong to a particular namespace for that instance of the document. Different instance documents may use different prefixes, but still conform to the schema.

In this example, the schema uses the prefixes **po** and **common**, but an instance document may use the prefixes **mypo** and **com**, and still conform to the schema.



This is an example of how the instance document might use a different prefix by declaring it differently on the xmlns:prefix attribute. Notice that the namespace itself is still the same, even though the prefix is different – **mypo** instead of **po**.



It is also possible to define a default namespace. This allows you to omit the prefix for elements that belong to that namespace.

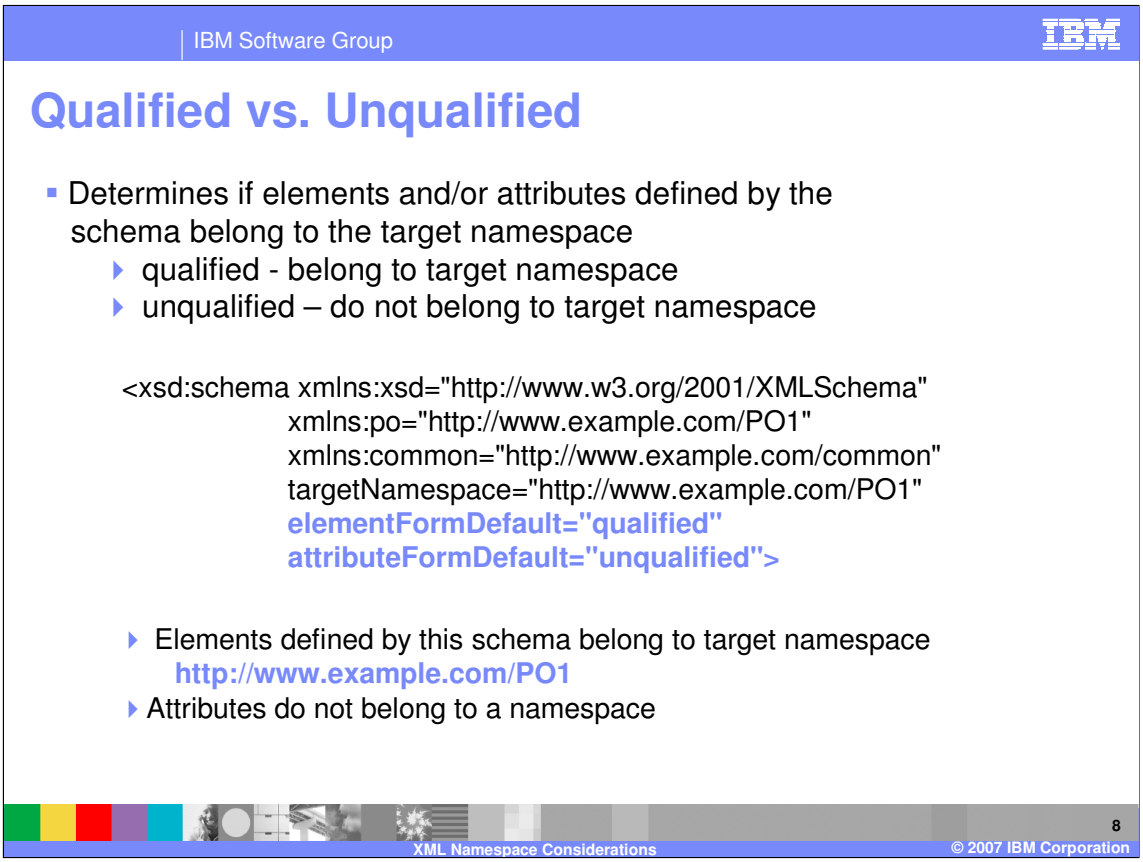
In this example, the **PONum** element is processed internally the same as the **PONum** element on the previous slide, even though there is no prefix here. This is because it belongs to the default namespace, which is declared using the same URI as the previous slide.

Target namespace

- Specifies the namespace defined by this schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:po="http://www.example.com/PO1"
            xmlns:common="http://www.example.com/common"
            targetNamespace="http://www.example.com/PO1"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
```


If the elements in the schema belong to a namespace, this namespace is identified using the **targetNamespace** attribute. Here is an example.



A schema can specify whether elements and/or attributes that are defined by the schema belong to the target namespace or not by using the **elementFormDefault** and **attributeFormDefault** attributes. “qualified” indicates that they do belong to the target namespace, “unqualified” means that they do not.

Typically, as in this example, elements are qualified and attributes are not.

Mapping and translation using a schema – with a target namespace

- Import the schema(s)
  - Namespaces that are declared with an xmlns attribute will automatically be added to the dictionary if needed
  - If schemas are nested, import those too
- Create the map
- Run the TRANSFORM command

If a schema has a target namespace, the steps for mapping and translation are still basically the same as with DTDs and other schemas:

-Import the schema – or multiple schemas if there are references to others

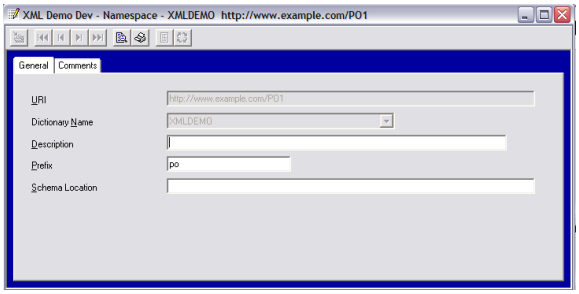
-Create the map

-Run the PERFORM TRANSFORM command

One difference you may notice when you import the schemas is that WDI adds namespace objects to the XML dictionary when the schemas are imported.

Namespace objects

- Import process scans schema for xmlns attribute
- If namespace object not already in dictionary, WDI will add it
- Namespace objects include:
  - URI
  - Prefix (can leave blank if this is to be used as default namespace)
  - Schema location



When you import an XML schema, WDI will scan it and look for xmlns attributes. If it finds any, WDI will add those namespaces to the XML dictionary if they do not already exist. If the namespace object DOES already exist, WDI does not replace the entry.

Namespace objects include:

-The URI, which identifies the namespace and must be unique within the XML dictionary.

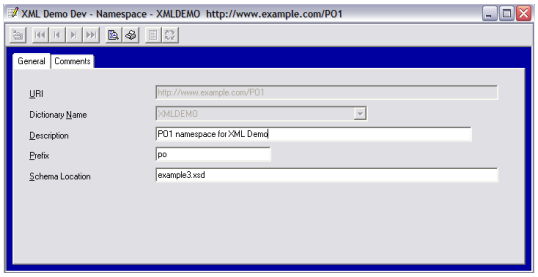
-An optional prefix. If you want to use this URI as the default namespace, you can leave the prefix blank.

-An optional schema location.

You can also specify an optional description, which is only used for documentation.

Namespace objects

- Namespace objects are used to:
  - Determine the prefix displayed in the Client
  - Determine the prefix for output XML data
  - Create schemaLocation and xmlns attributes (more on that later)
- The prefix in the Namespace object DOES NOT have to match the prefix in either the schema or the XML data
  - But it simplifies problem determination if it does!



Namespace objects have several purposes in WDI. They are used to help determine:

•The prefix for the tree views in the schema and map editors.

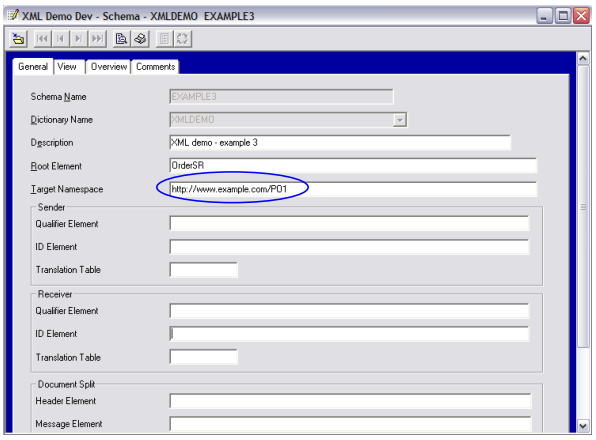
•The prefix for the output XML data

•The values for schemaLocation and xmlns attributes

The prefix can be different from the one used in the schema and/or XML data. But it is best to keep it as consistent as possible to simplify problem determination.

Schema editor – General tab

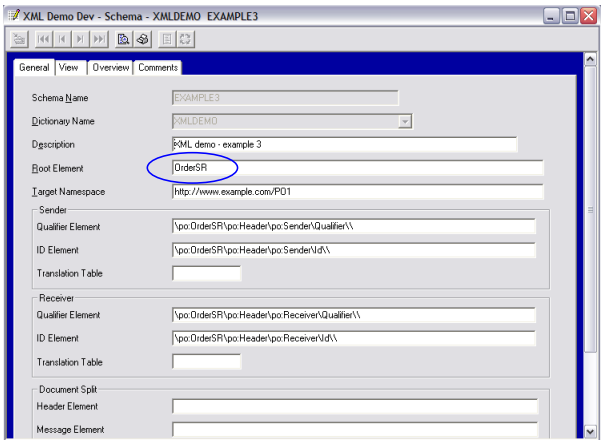
- Target namespace filled in automatically by import



This slide shows the General tab on the schema editor. The Target Namespace field was filled in automatically when the schema was imported.

Schema editor – General tab

- DO NOT include prefix on root element

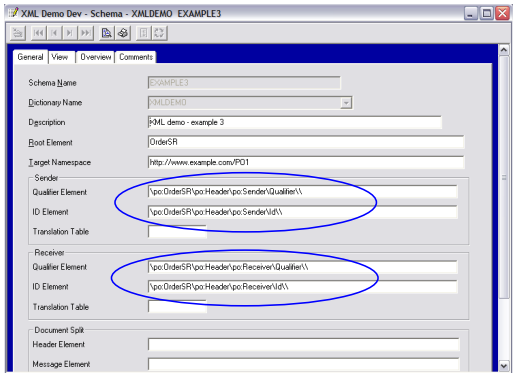


As with DTDs and other schemas, you need to specify the Root Element so WDI knows which element to use to start constructing the tree.

Even if the schema uses namespaces, you DO NOT include the prefix for this field.

Schema editor – General tab

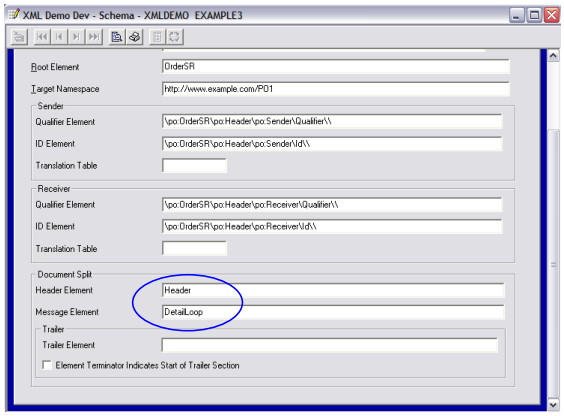
- DO include prefix on qualified elements for sender/receiver element paths



When you specify the paths for the Sender and Receiver ID and Qualifier elements, these should include the prefixes for any qualified elements. So if the elements appear in the tree view with the prefix, they should also have the prefix here. When you use the right-click logic from the tree view to set these paths, the prefixes are automatically included for qualified elements.

Schema editor – General tab

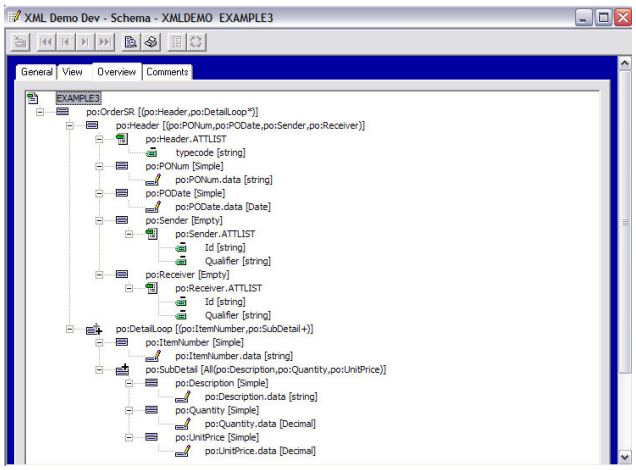
- DO NOT include prefix on XML split elements



For the Document Split elements, you DO NOT include the prefix, even if the element is qualified in the XML schema.

Schema editor – Overview tab

- Qualified elements and attributes display appropriate prefix

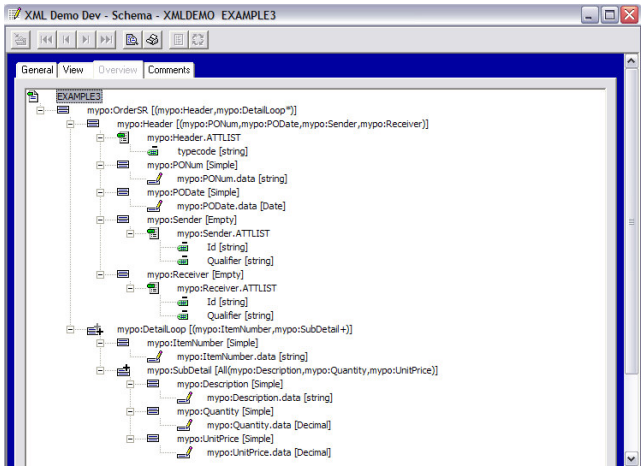


This is an example of the tree view for the XML schema. The prefix, “po” in this case, is taken from the namespace object.

You may notice that the attributes do not have a prefix. That is because in this example, the elements are qualified, but the attributes are not.

Schema editor – Overview tab

- Note how element prefixes change if you change the prefix on the Namespace object



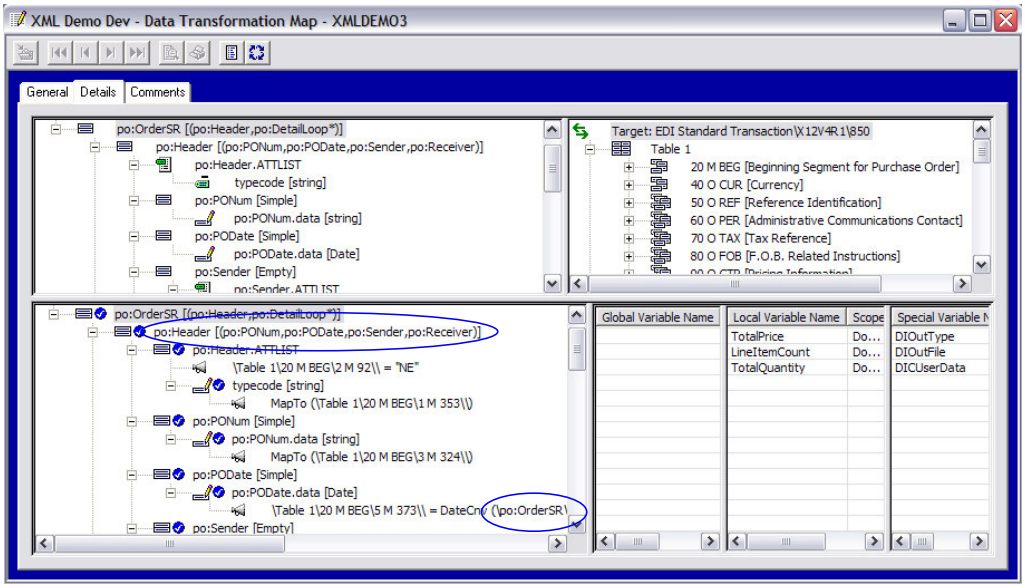
If the prefix in the namespace object changes, the prefix that appears in the tree view also changes the next time you open the XML schema. Notice how in this case the “po” prefix changed to “mypo”.

Mapping from a schema source document – with namespaces

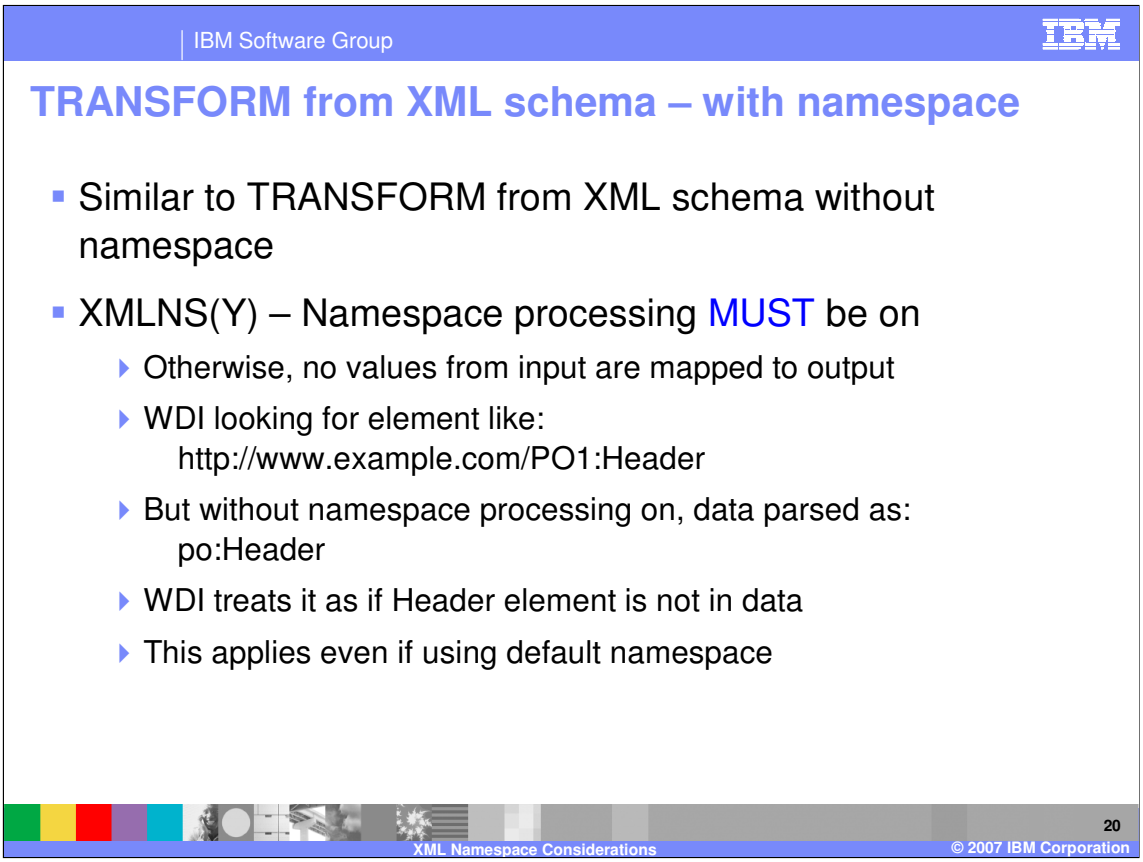
- Similar to mapping from a schema without namespaces
- Elements displayed in the map similar to the way they appear on the schema editor Overview tab
  - Includes prefixes as appropriate

Mapping is similar whether the schema uses namespaces or not. The most noticeable difference for namespaces is that the map editor shows the prefixes for the qualified elements and attributes, similar to the way the schema Overview tab displayed them.

Schema map – source document



You can see in this example that the prefix from the namespace object is included for the elements – both in the tree view and when the path names are used in the commands.



When using namespaces in the source XML schema, the PERFORM TRANFORM is also similar. Using the keyword XMLNS(Y) is generally recommended any time you are using XML schemas, but if the schema uses XML namespaces, XMLNS(Y) is REQUIRED.

This is because the map uses the entire namespace URI to define the qualified elements and attributes – not the prefix. For example, the fully-qualified Header element that is saved in the map might be something like “http://www.example.com/PO1:Header”. This allows WDI to process the data regardless of what prefix is used on a particular instance document – as a namespace-aware application is supposed to.

However, you need to tell WDI that it should act as a namespace-aware application. Otherwise, it will not convert the shorthand prefix to the namespace URI, and would parse the element name as “po:Header” instead. This would result in a mismatch between the element name in the map, and the element name parsed from the data, and the Header element would not get mapped correctly.

Note that you may need to specify XMLNS(Y), even if you do not see any prefixes on the element and attribute names in the data. If the schema uses the default namespace, you do not see the prefixes in the data, but the elements and attributes are still treated as fully-qualified names in the map.

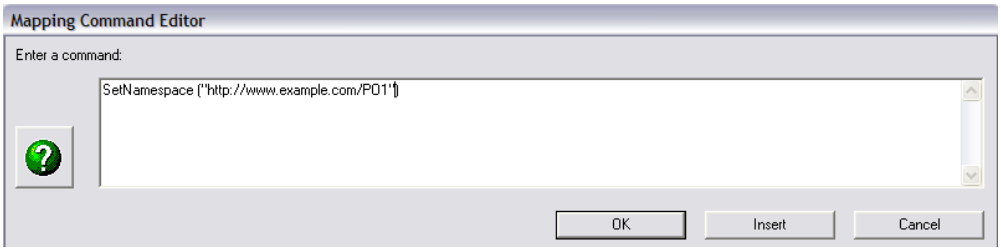
Mapping to a schema target document – with namespaces

- Similar to mapping to a schema without namespaces
- Elements displayed in the map similar to the way they appear on the schema editor Overview tab
  - Includes prefixes as appropriate
- A couple of new mapping commands

Mapping to a target document that is based on an XML schema is also similar, whether the schema uses namespaces or not. Again, the map editor displays the prefixes for qualified elements and attributes. However, there are a couple of new mapping commands that only apply when the target XML document uses namespaces.

SetNamespace command

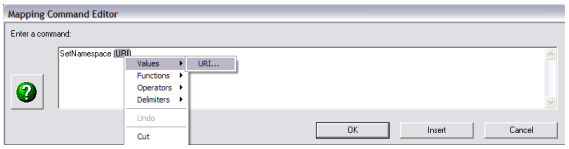
- SetNamespace(URI)
  - Creates: xmlns:prefix attribute in output
  - Uses the prefix, URI defined in the Namespace object
  - Example:
    xmlns:po="http://www.example.com/PO1"
  - Note: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    is automatically generated if needed



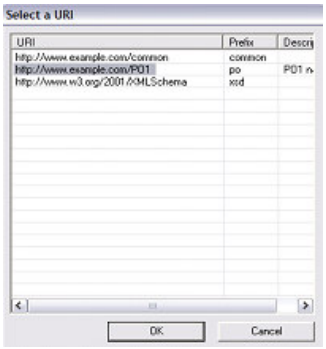
The SetNamespace command creates the xmlns:prefix attribute in the output. You pass the URI as a parameter, and the prefix is filled in from the namespace object in the XML dictionary.

Setting URIs in mapping commands

- Hint: Right-click on URI to show list of available namespace URIs



- Which will display a list of all URIs in the dictionary

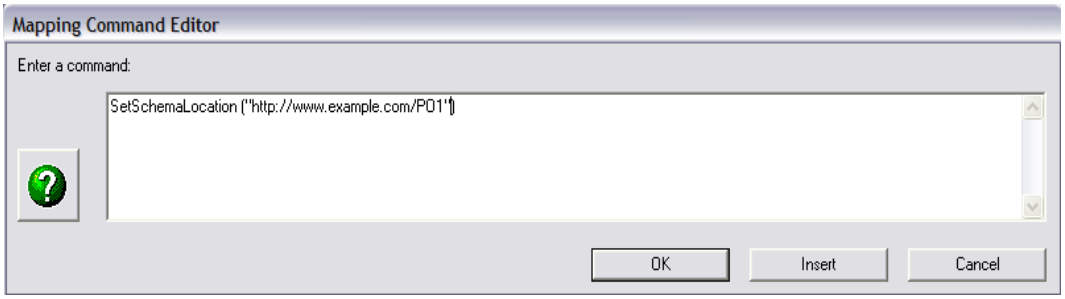


When you are setting the URI in the mapping commands, the easiest way is to right-click on the "URI" parameter to display a list of all URIs in the dictionary. Then you can select the URI from the list, instead of trying to type the full URI and hoping it matches exactly.

SetSchemaLocation command

- SetSchemaLocation(URI)
  - Creates: xsi:schemaLocation attribute in output
  - Uses the URI, Location defined in the Namespace object
  - Example:

xsi:schemaLocation="http://www.example.com/PO1 example3.xsd"

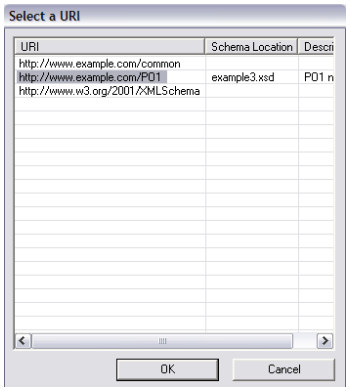


The SetSchemaLocation mapping command is used to create the schemaLocation attribute on the root element. You would use this when the target document is based on a schema, you want to identify the schema name in the XML output, and the XML schema DOES use a target namespace. When you use this command, you should also have a corresponding SetNamespace command for the URI.

The URI for the schemaLocation is passed as a parameter, and the location is read from the namespace object.

Setting URI in SetSchemaLocation commands

- Also allows right-click on URI to show list of available namespace URIs
- List of URIs includes schema location



This mapping command also allows you to right-click on the URI to show the list of available URIs. However, for the setSchemaLocation command the list includes the schema location instead of the prefix.

Keywords for TRANSFORM to XML schema – with namespaces

- Again, no special PERFORM keywords for transforming to an XML schema-based target document with namespaces
- Target output processing determined by map, rules, trading partner info, etc.

Like with other variations of XML output we have discussed, there are no special PERFORM TRANSFORM keywords needed to generate XML output based on a schema with namespaces. The target output processing is determined by the map, rules, trading partner information, etc.

Summary

- Namespaces help to resolve name conflicts when multiple schemas are used to define a document
- Mapping and transformation are similar to XML documents that do not use namespaces
- There are some special considerations when using XML namespaces with WDI, including new mapping commands and keywords

In summary, namespaces help to resolve name conflicts when multiple schemas are used to define a document. We talked a bit about some general concepts, and what it means for an application to be “namespace-aware”.

For the most part, mapping and transformation are similar to XML documents that do not use namespaces. However, there are some special considerations, including new mapping commands and keywords. If the source XML document uses a schema with namespaces, the use of the XMLNS(Y) keyword is particularly important. This way WDI knows that it should parse the XML data as a namespace-aware application.

Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	WMQ	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.