

# Working with Data

# Objectives

After completing this lesson, you should be able to:

- Describe the role of XML, XSD, XPath, XQuery, and XSLT in the way Oracle SOA Suite 12c handles data
- Use the XSLT Mapper to create XSL transformations in a Mediator
- Describe how the XQuery Mapper can be used to create XQuery transformations in a Mediator

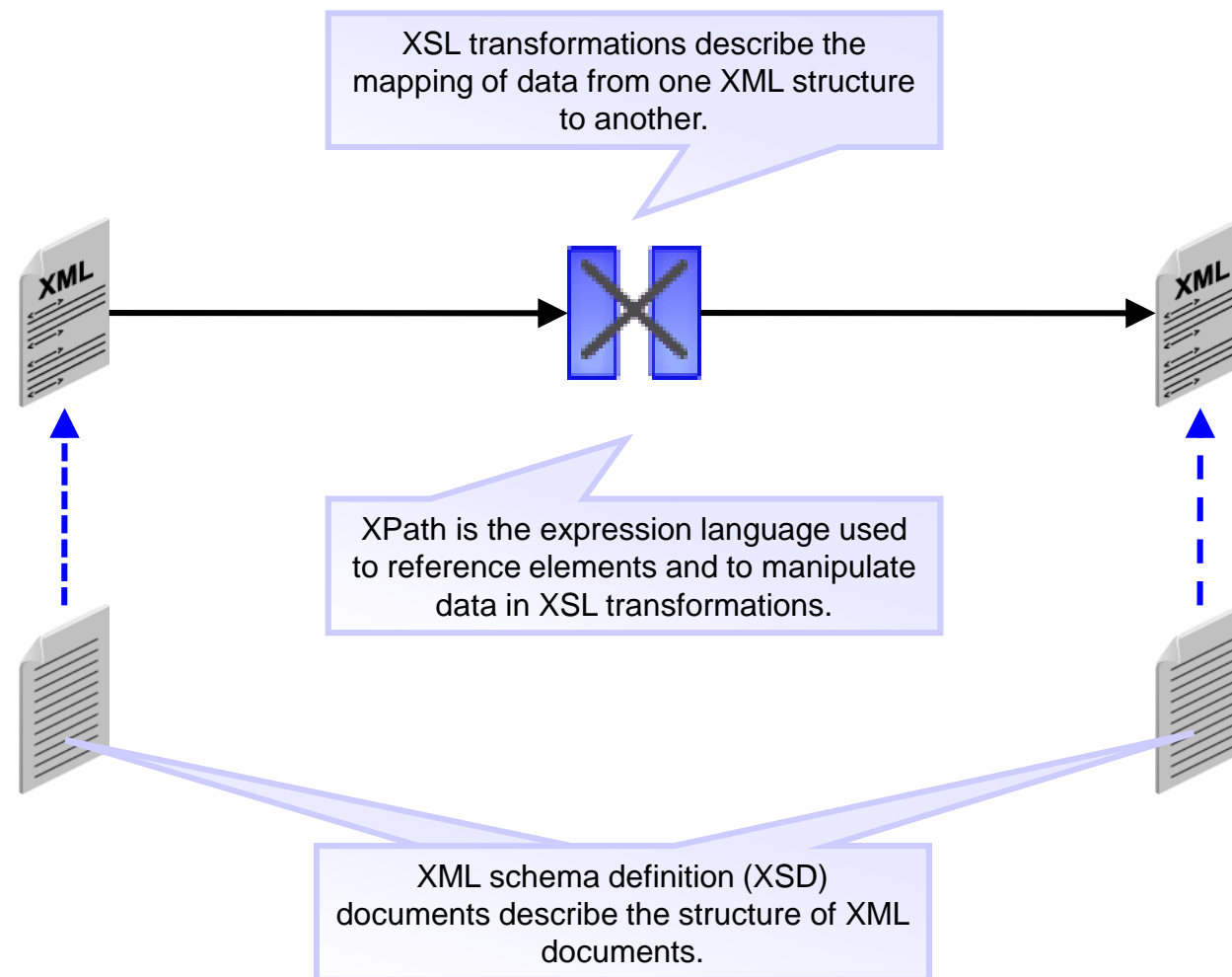


# Agenda

- XSLT Transformations in Mediator
- XQuery Transformations in Mediator



# Data Standards



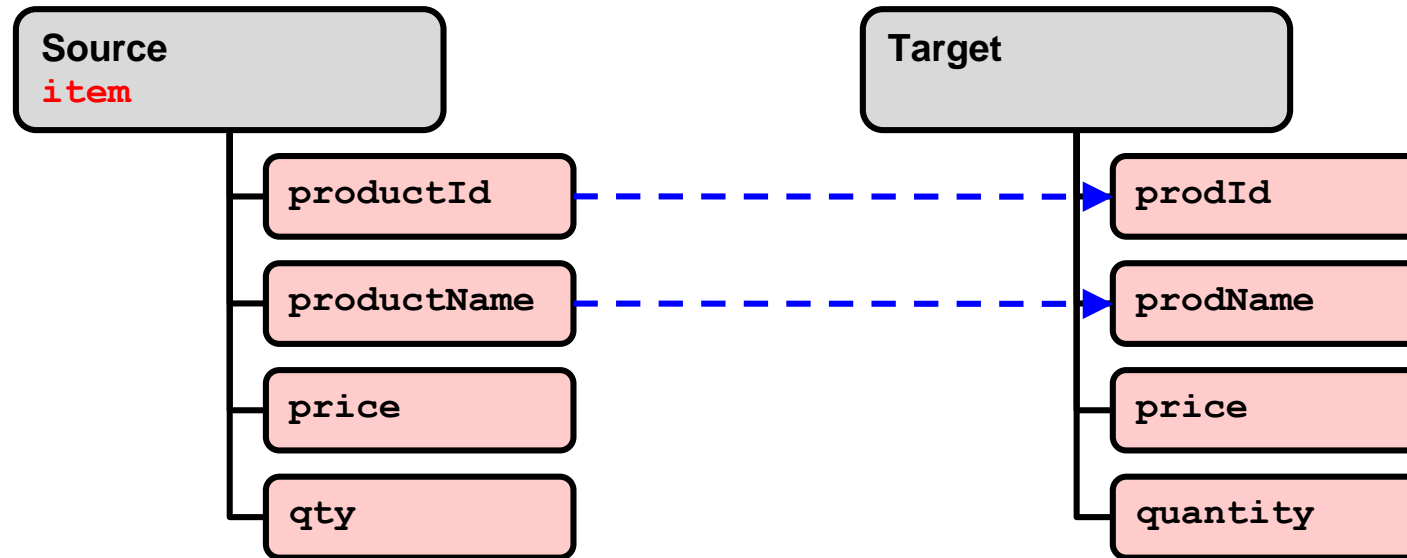
# Role of XSD Attributes

```
<ns0:item>
  <ns0:productId>SKU301</ns0:productId>
  <ns0:name>Music Player 1Gb</ns0:productName>
  <ns0:price>45</ns0:price>
  <ns0:quantity>3</ns0:quantity>
</ns0:item>
```

*XSD describes XML.*

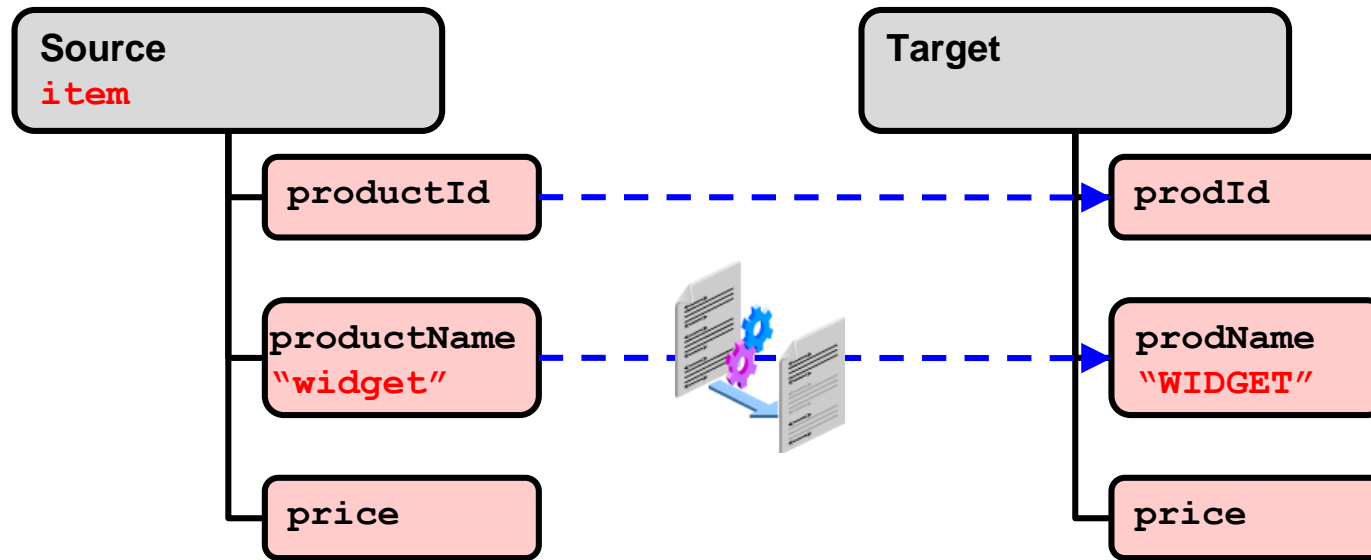
```
<xsd:complexType name="OrderItemType">
  <xsd:sequence>
    <xsd:element name="productId" type="xsd:string" minOccurs="1"/>
    <xsd:element name="name" type="xsd:string" minOccurs="1"/>
    <xsd:element name="price" type="xsd:decimal" minOccurs="1"/>
    <xsd:element name="quantity" type="xsd:int" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

# XSL Transformations



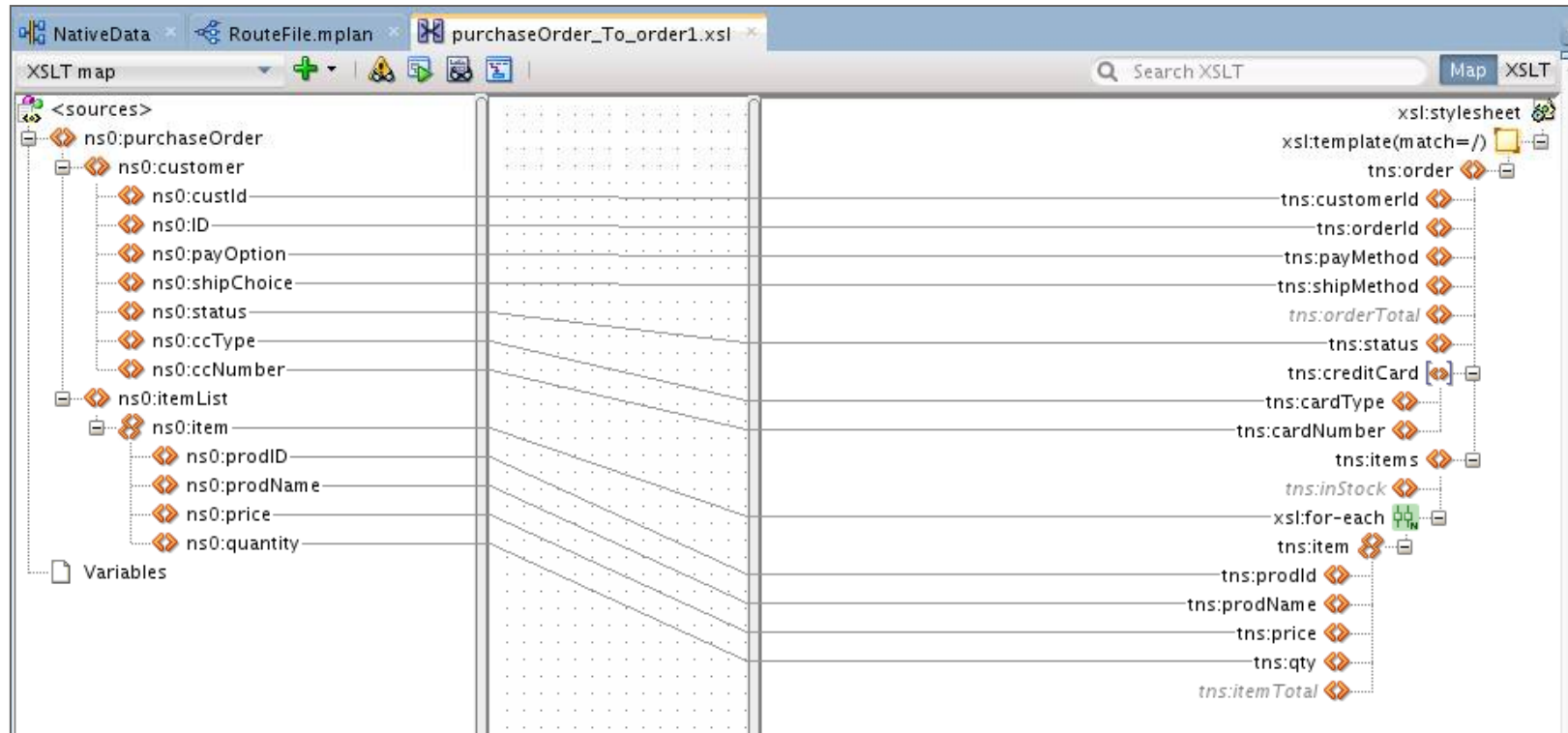
```
<ns1:item>
  <ns1:prodId>
    <xsl:value-of select="impl:productId"/>
  </ns1:prodId>
  <ns1:prodName>
    <xsl:value-of select="impl:productName"/>
  </ns1:prodName>
</ns1:item>
```

# Using XPath Functions



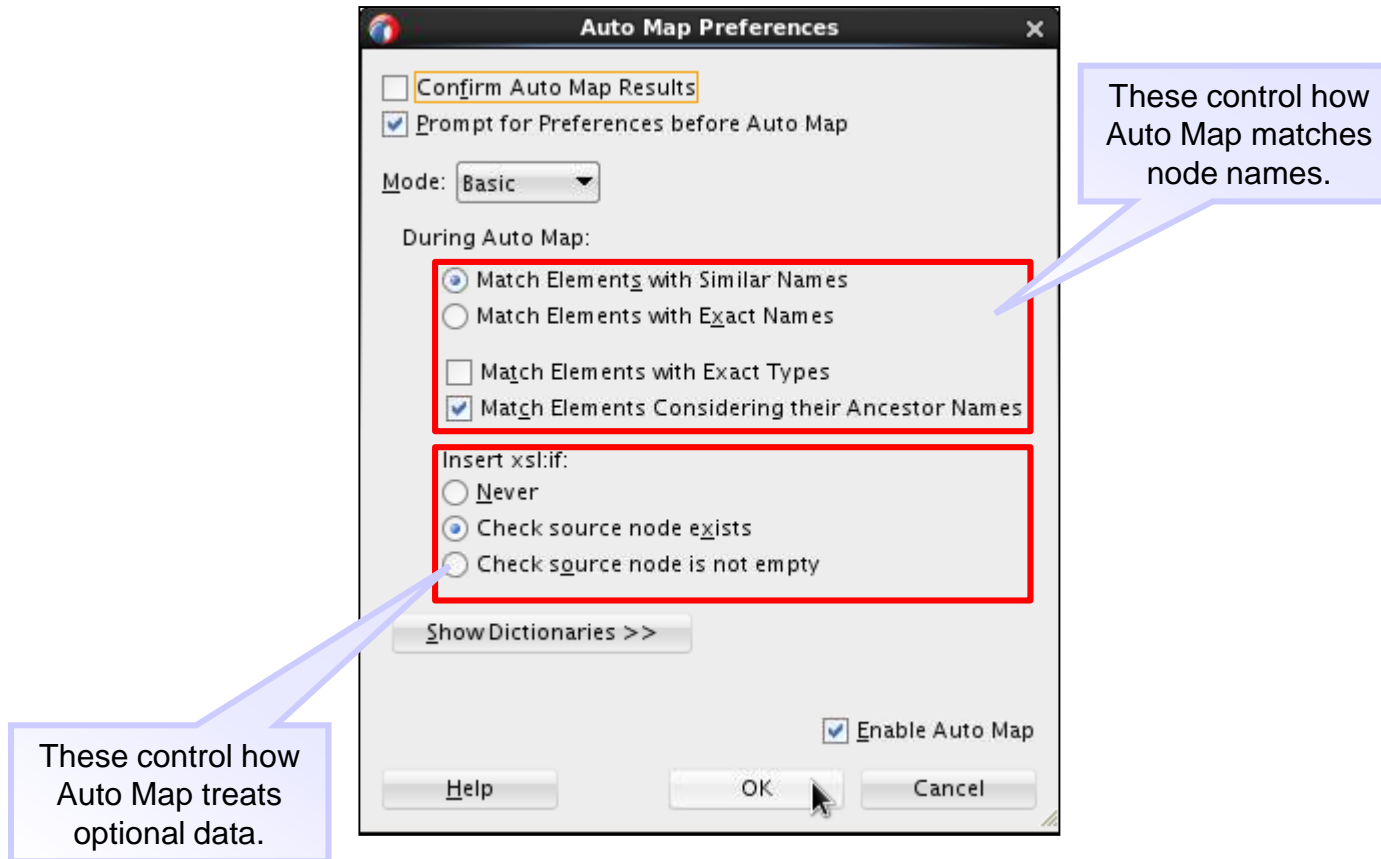
```
<ns1:item>
  <ns1:prodId>
    <xsl:value-of select="impl:productID"/>
  </ns1:prodId>
  <ns1:prodName>
    <xsl:value-of select="xp20:upper-case(impl:productName)"/>
  </ns1:prodName>
</ns1:item>
```

# Using the Mapper in JDeveloper





# Auto Map



# XPath Functions

The screenshot displays the Oracle JDeveloper XSLT map editor. The main workspace shows an XSLT map with source and target nodes. A red box highlights a function icon (xp20:upper-case) being dragged from the Component Palette to the target node. A callout bubble points to this icon with the text: "Functions appear as icons." Another callout bubble points to the Component Palette with the text: "The Component Palette offers drag-and-drop access to the XPath function library." The Component Palette is located on the right side of the interface and lists various XPath functions under the "General XPath" category. The bottom of the interface shows the "Expression Context" panel, which displays the XPath expression being edited: `XPath ==> tns:prodName`. Below this, the "XPath Expression" field contains the function call: `xp20:upper-case (ns0:prodName )`. Examples of XPath expressions are provided: `$var1`, `$pfx:param1`, `'abc'`, and `"abc"`. A note indicates that pressing `Ctrl + Space` invokes the XPath Building Assistant.

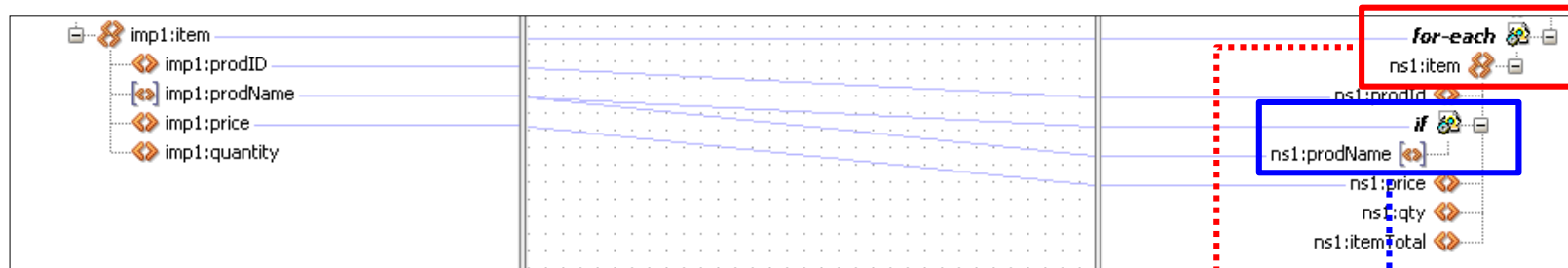
Functions appear as icons.

The Component Palette offers drag-and-drop access to the XPath function library.

Expression Context  
Context  
Parameters  
Variables  
Context Nodes  
ns0:item

XPath ==> tns:prodName  
XPath Expression:  
xp20:upper-case (ns0:prodName )  
Examples: \$var1, \$pfx:param1, 'abc', "abc"  
Press Ctrl + Space for Invoking XPath Building Assistant.

# Optional and Repeating Data



```
<xsd:element name="item" type="tns:itemType" maxOccurs="unbounded" />  
<xsd:element name="prodName" type="xsd:string" minOccurs="0"/>
```

# Testing Transformations

The screenshot displays the Oracle XML Editor interface. The top half shows two XML files side-by-side: 'purchaseOrder\_To\_order1-Source.xml' and 'purchaseOrder\_To\_order1-Target.xml'. The bottom half shows the 'XSLT map' panel, which contains a tree view of the source and target namespaces and a central mapping area. A callout box points to the source and target panels, stating: 'Source and destination panels display sample data before and after transformation.'

**Source XML (purchaseOrder\_To\_order1-Source.xml):**

```
<?xml version="1.0" encoding="UTF-8" ?>
<purchaseOrder xmlns="http://TargetNamespace.com/ReadF">
  <customer>
    <custId>custId1</custId>
    <ID>ID2</ID>
    <payOption>payOption3</payOption>
    <shipChoice>shipChoice4</shipChoice>
    <status>status5</status>
    <ccType>ccType6</ccType>
    <ccNumber>ccNumber7</ccNumber>
  </customer>
  <itemList>
    <item>
      <prodID>prodID8</prodID>
      <prodName>prodName9</prodName>
      <price>10</price>
      <quantity>11</quantity>
    </item>
    <item>
      <prodID>prodID12</prodID>
      <prodName>prodName13</prodName>
      <price>14</price>
      <quantity>15</quantity>
    </item>
  </itemList>
</purchaseOrder>
```

**Target XML (purchaseOrder\_To\_order1-Target.xml):**

```
<?xml version="1.0" encoding="UTF-8" ?>
<tns:order xmlns:jca="http://xmlns.oracle.com/pcbpel/v">
  <tns:customerId>custId1</tns:customerId>
  <tns:orderId>ID2</tns:orderId>
  <tns:payMethod>payOption3</tns:payMethod>
  <tns:shipMethod>shipChoice4</tns:shipMethod>
  <tns:status>status5</tns:status>
  <tns:creditCard>
    <tns:cardType>ccType6</tns:cardType>
    <tns:cardNumber>ccNumber7</tns:cardNumber>
  </tns:creditCard>
  <tns:items>
    <tns:item>
      <tns:prodId>prodID8</tns:prodId>
      <tns:prodName>PRODNAME9</tns:prodName>
      <tns:price>10</tns:price>
      <tns:qty/>
    </tns:item>
    <tns:item>
      <tns:prodId>prodID12</tns:prodId>
      <tns:prodName>PRODNAME13</tns:prodName>
      <tns:price>14</tns:price>
      <tns:qty/>
    </tns:item>
  </tns:items>
</tns:order>
```

**XSLT map:**

- Source namespace: ns0
- Target namespace: tns
- Mapping: ns0:purchaseOrder to tns:order
- Mapping: ns0:customer to tns:customer
- Mapping: ns0:custId to tns:customerId
- Mapping: ns0:ID to tns:orderId
- Mapping: ns0:payOption to tns:payMethod
- Mapping: ns0:shipChoice to tns:shipMethod
- Mapping: ns0:status to tns:status
- Mapping: ns0:ccType to tns:creditCard
- Mapping: ns0:ccNumber to tns:cardNumber
- Mapping: ns0:itemList to tns:items
- Mapping: ns0:item to tns:item
- Mapping: ns0:prodID to tns:prodId
- Mapping: ns0:prodName to tns:prodName
- Mapping: ns0:price to tns:price
- Mapping: ns0:quantity to tns:qty

# Quiz



The Mediator can apply XSLT mappings to message structures that are processed in routing rules.

- a. True
- b. False



# Agenda

- XSLT Transformations in Mediator
- XQuery Transformations in Mediator



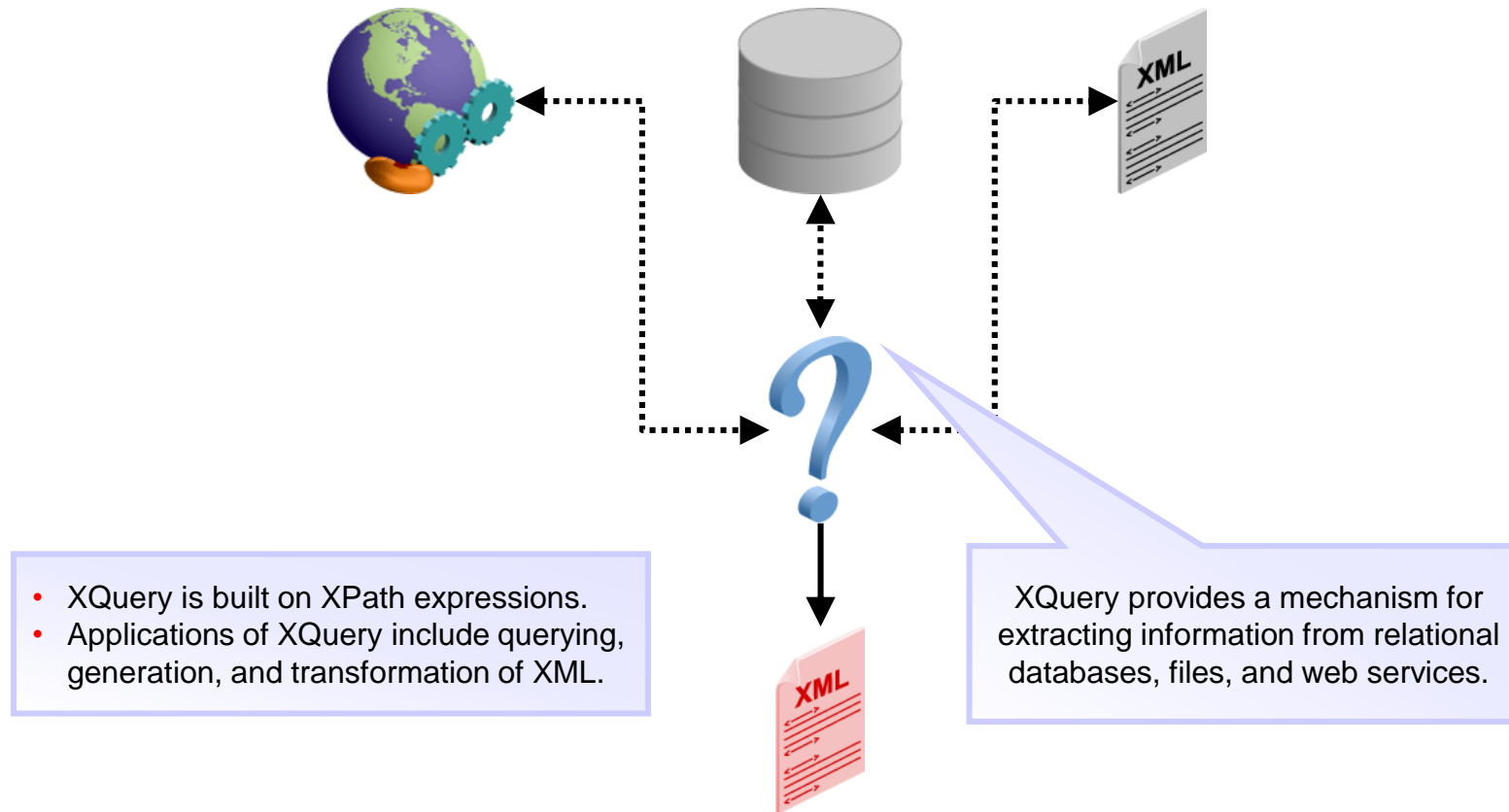
# XQuery: Introduction

According to the W3C standards body that oversees the development of the XQuery standard:

- XQuery is a standardized language for *combining documents, databases, web pages*, and almost anything else.
- XQuery is replacing proprietary middleware languages and web application development languages. XQuery is *simpler to work with and easier to maintain than many other alternatives*.

XQuery is designed to work with the XML data model and be a comprehensive query language for data that is expressed in XML.

# XQuery

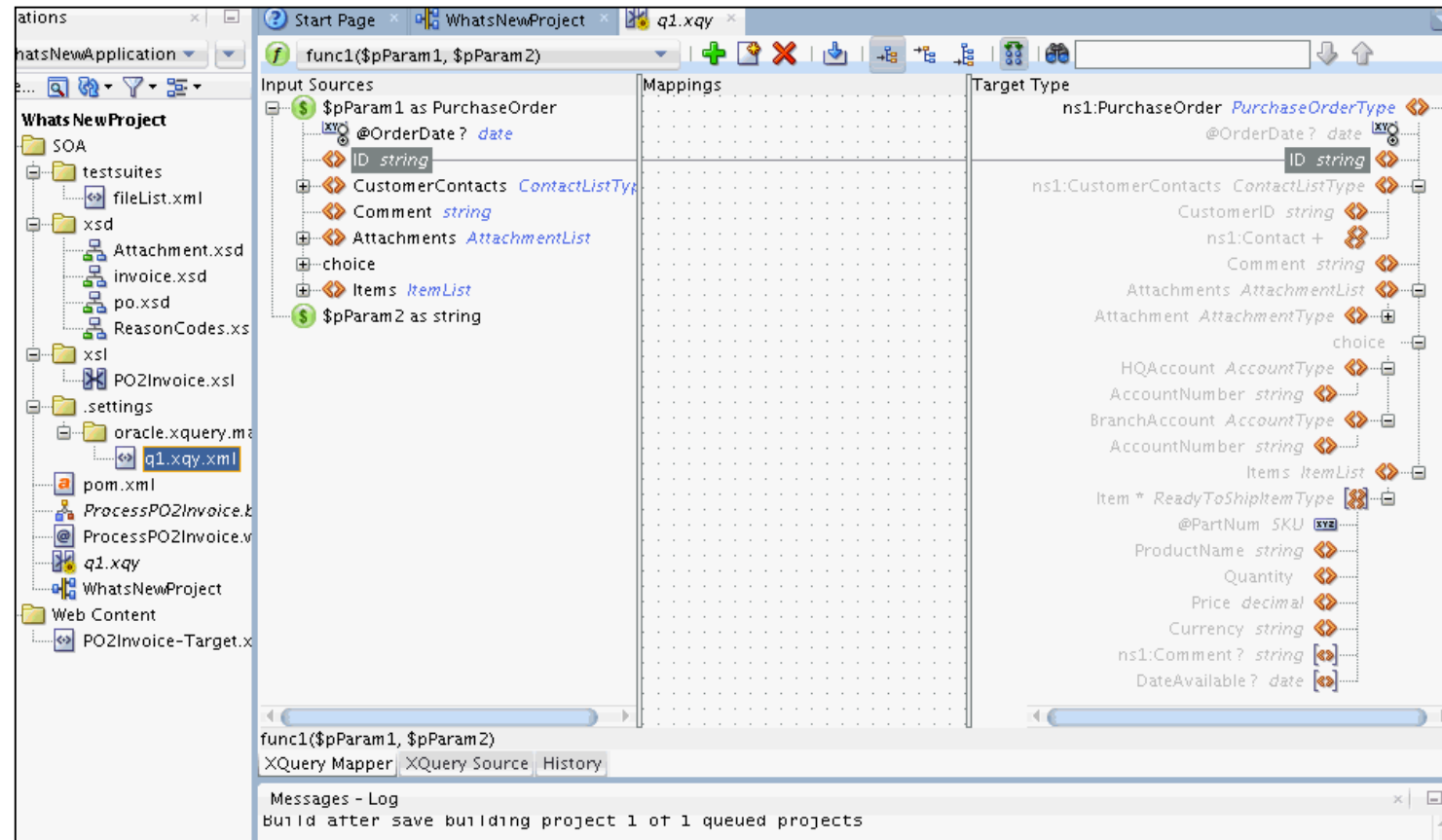




# XSLT Versus XQuery

- Similarities:
  - XPath
  - Data model
  - Functions and operators
- Differences:
  - Syntax
    - XQuery is similar to SQL.
    - XSLT stylesheets use XML syntax.
  - Performance
    - XSLT loads the entire input document in memory.
    - XQuery loads only the objects that need to be used by the current statement.

# Using XQuery Transformations



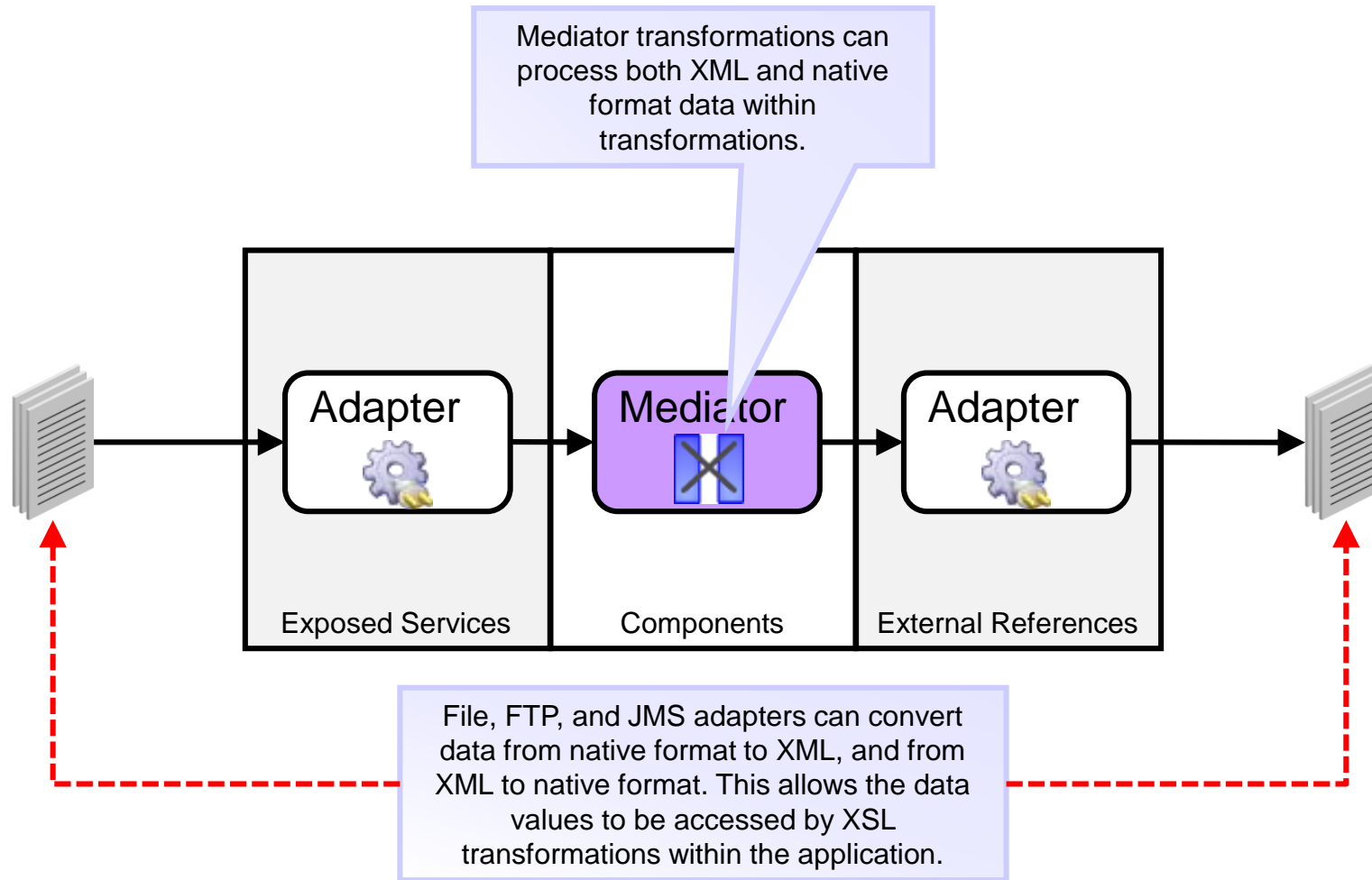
12<sup>c</sup>

# Agenda

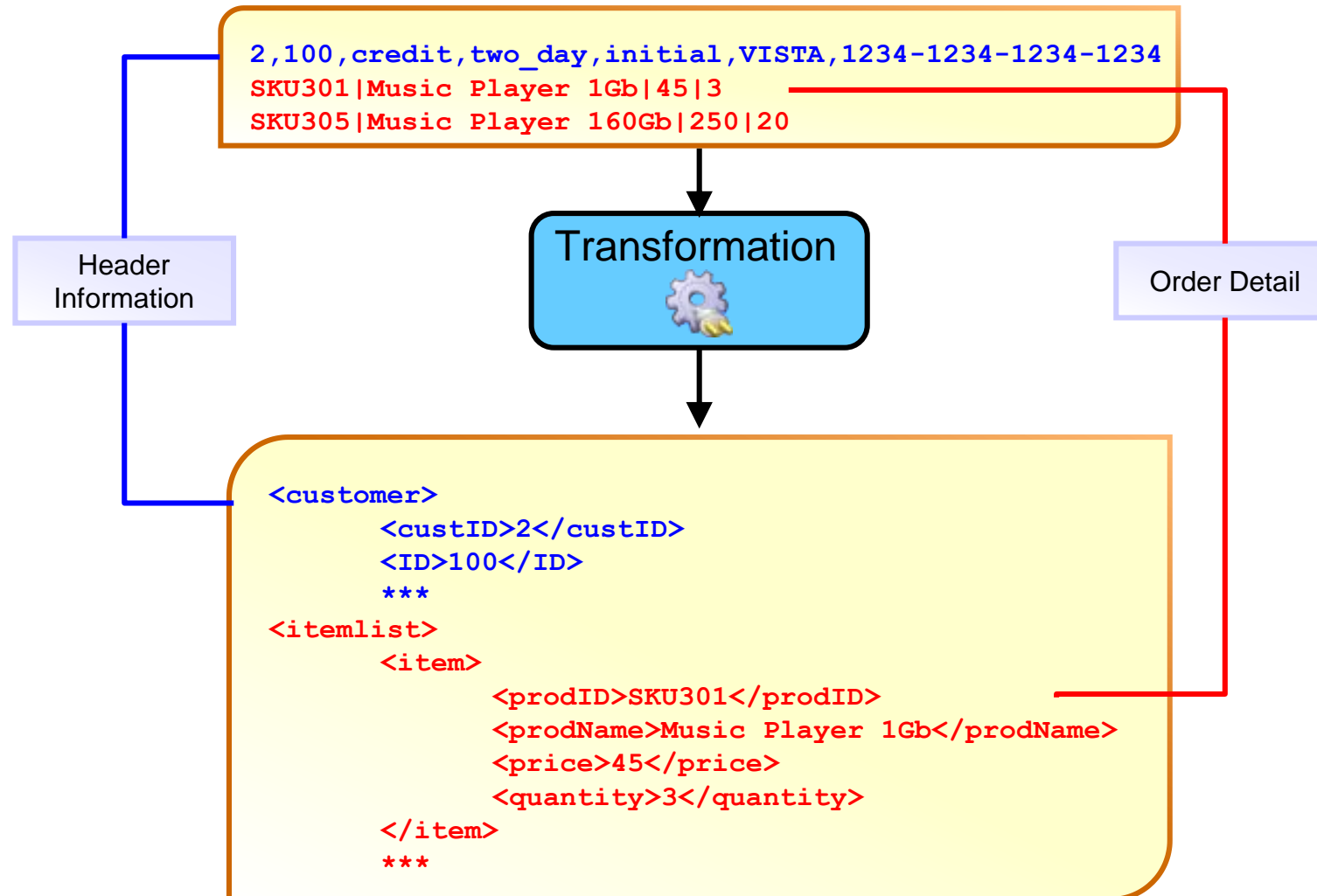
- XSLT Transformations in Mediator
- XQuery Transformations in Mediator
- **Adapters and Native Format Data**



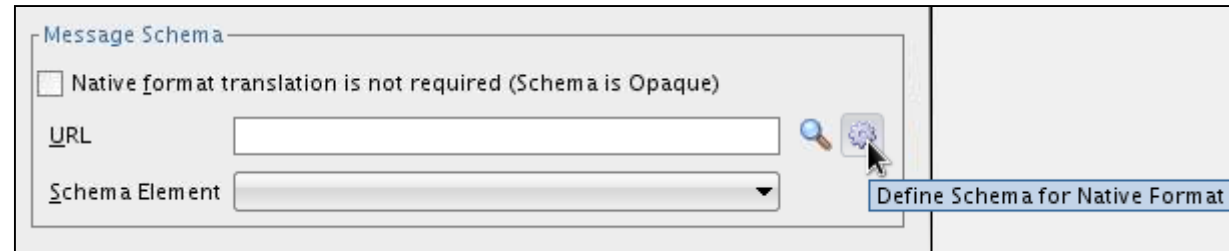
# Working with Native Format Data



# Native Data Transformation



# Invoking the Native Format Builder



Message Schema

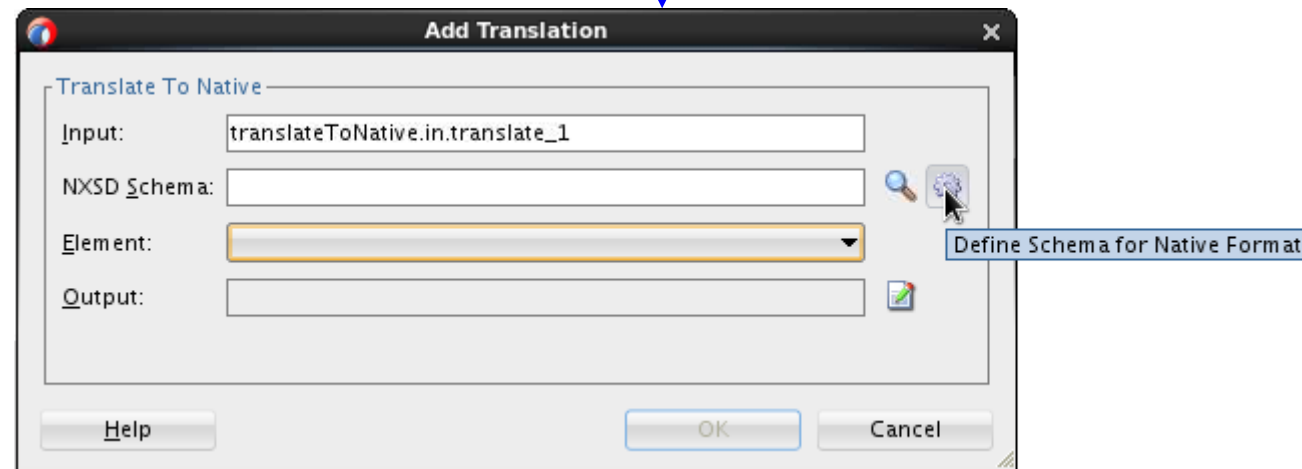
☐ Native format translation is not required (Schema is Opaque)

URL

Schema Element

Define Schema for Native Format

Invoke the Native Format Builder from either the **Adapter Wizard** or via the **Mediator Translate to Native** option.



Add Translation

Translate To Native

Input:

NXSD Schema:

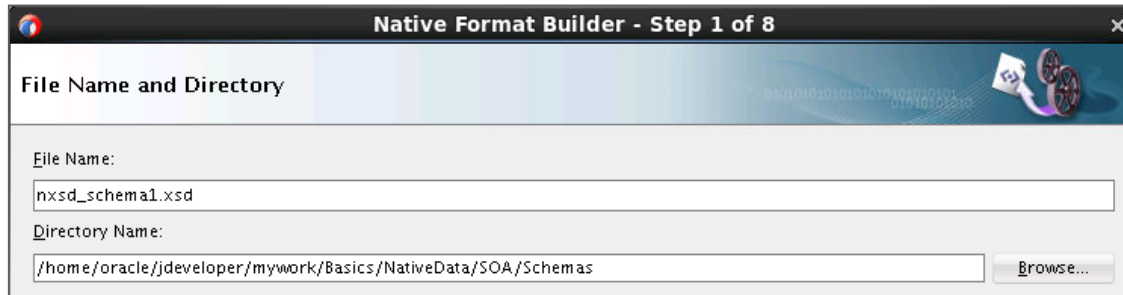
Element:

Output:

Define Schema for Native Format

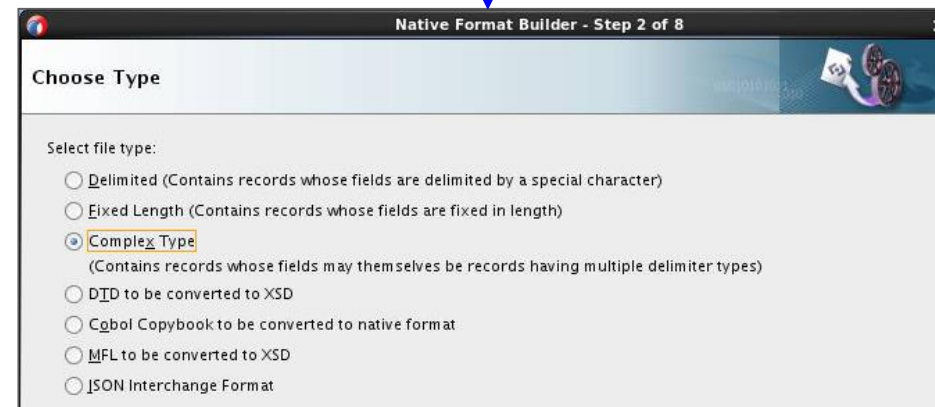
Help OK Cancel

# Specifying File Name and Native Data Format



The screenshot shows the 'Native Format Builder - Step 1 of 8' window. The title bar includes the Oracle logo and the text 'Native Format Builder - Step 1 of 8'. The window has a header bar with the title 'File Name and Directory'. Below the header, there are two text input fields. The first field is labeled 'File Name:' and contains the text 'nxsd\_schema1.xsd'. The second field is labeled 'Directory Name:' and contains the path '/home/oracle/jdeveloper/mywork/Basics/NativeData/SOA/Schemas'. To the right of the 'Directory Name' field is a 'Browse...' button.

Specify the **name of the file to create**,  
and the **type of native data** to process.



The screenshot shows the 'Native Format Builder - Step 2 of 8' window. The title bar includes the Oracle logo and the text 'Native Format Builder - Step 2 of 8'. The window has a header bar with the title 'Choose Type'. Below the header, there is a section titled 'Select file type:' followed by a list of radio button options. The options are: 'Delimited (Contains records whose fields are delimited by a special character)', 'Fixed Length (Contains records whose fields are fixed in length)', 'Complex Type (Contains records whose fields may themselves be records having multiple delimiter types)', 'DID to be converted to XSD', 'Cobol Copybook to be converted to native format', 'MFL to be converted to XSD', and 'JSON Interchange Format'. The 'Complex Type' option is selected and highlighted with a yellow box.

# Specifying a Sample File

**Native Format Builder - Step 3 of 5**

### File Description

Specify name of file that you want to sample

File name:

Number of rows to skip:

Number of rows to sample:  ☐ All rows

Number of data rows to process:  ☒ All rows

Character set:

Target namespace:

Root element:

**Select schema and payload validation flags**

☐ Validate Schema ☐ Validate Payload at Top level ☐ Validate Payload at Field level

**Select Complex LookAhead Processing Flags**

☐ Report LookAhead Error

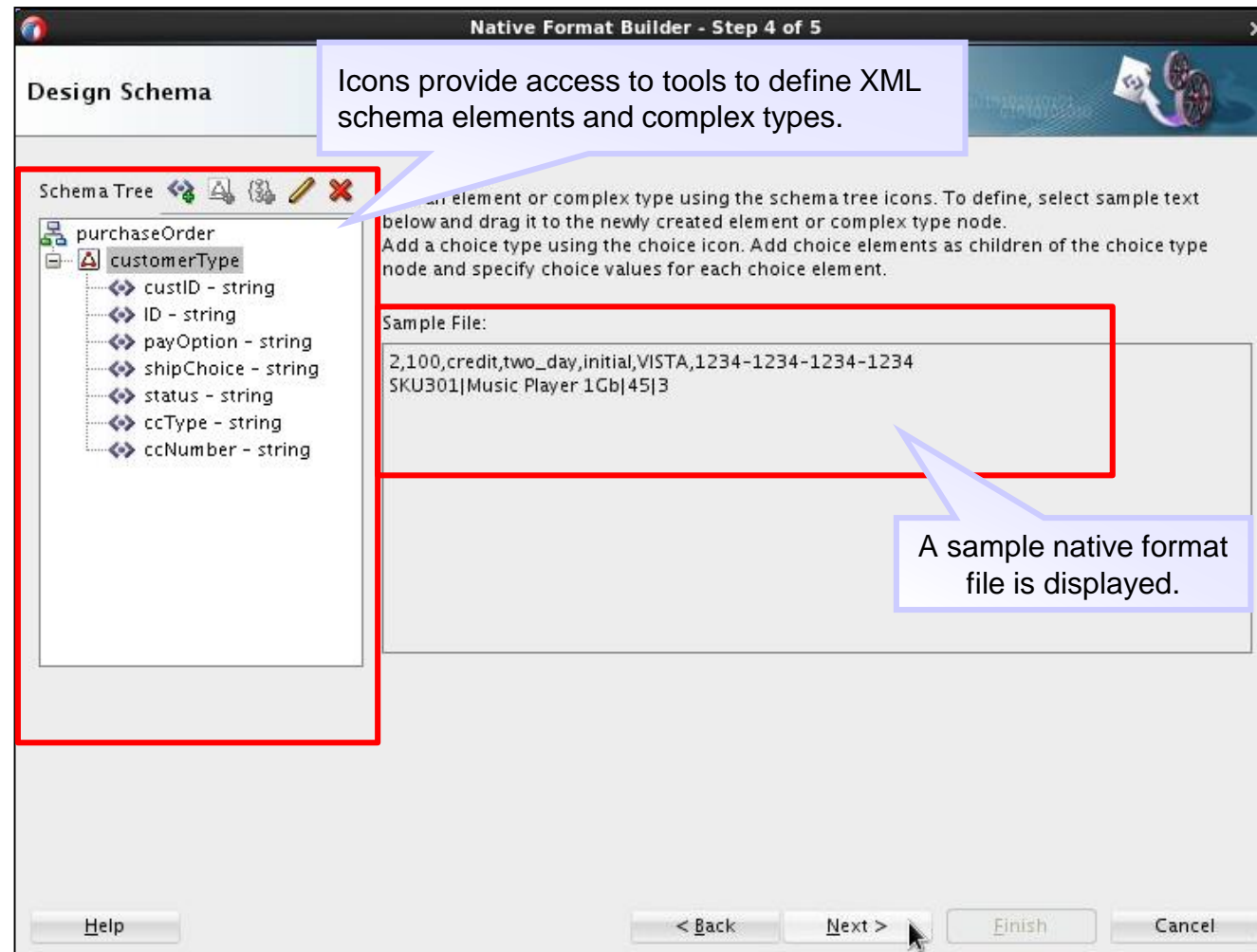
File: home/oracle/.../purchaseOrder.dat

```
2,100,credit,two_day,initial,VISTA,1234-1234-1234-1234
SKU301|Music Player 1Gb|45|3
SKU305|Music Player 120Gb|250|20
```

Specify a sample file and specify how it should be interpreted.



# Defining a Schema for a Native Format



# Summary

In this lesson, you should have learned how to:

- Describe the role of XML, XSD, XPath, and XSLT in the way Oracle SOA Suite 12c handles data
- Use the XSLT Mapper to create XSL transformations in a Mediator
- Test XSL transformations



# Practice 4 Overview

This practice covers the following topics:

- Using the XSLT Mapper

# Practice 4 Overview

