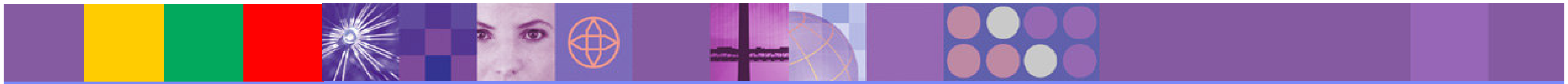
2004 WDI / WBIC Customer Conference

Global Business Transformation

WDI XML Considerations

Fritz Fahrenback

WebSphere. software



business software

Introduction

Fritz Fahrenback

- WebSphere Data Interchange development
- fritzf@us.ibm.com

Objectives

- Explain XML basic concepts (XML 101)
  - Document Structure
  - DTDs
  - Schemas
  - Namespaces
- Describe steps to map/translate to or from XML
- Demonstrate mapping and translation of an EDI transaction to an XML schema

What is XML ?

- XML is eXtensible Markup Language
- Becoming a common way of exchanging data with other parties and applications
- A W3C "Recommendation"
  - XML and related information at: http://www.w3.org/
- Actually a meta-markup language
  - Does not define semantics or tags
  - Specifies rules for defining and using tags to structure data
  - You define the tags!

Comparison to HTML

- XML is a subset of SGML
  - 80-20 rule: eliminates some of the more complex but rarely used SGML function
- HTML is also derived from SGML
  - XML has a similar look and feel
- However - many important differences between XML, HTML

XML vs. HTML

XML	HTML
Tags are defined by a specific implementation	Defines the tags
Tags define the structure and meaning of the data	Tags define layout and formatting of the data
Elements and attributes ARE case sensitive	Elements and attributes are NOT case sensitive
All start tags MUST have a corresponding end tag	Some tags do not have end tags
Tags MUST be nested properly	Processing does not always enforce nesting hierarchy

XML Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE personnel SYSTEM "personal.dtd">

<personnel>

  <person id="Big.Boss" >
    <name><family>Boss</family> <given>Big</given></name>
    <email>chief@foo.com</email>
    <link subordinates="one.worker two.worker three.worker four.worker
       five.worker"/>
  </person>

  <person id="one.worker">
    <name><family>Worker</family> <given>One</given></name>
    <email>one@foo.com</email>
    <link manager="Big.Boss"/>
  </person>
   ......

</personnel>
```

XML Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE personnel SYSTEM "personal.dtd">

<personnel>

 <person id="Big.Boss" >
   <name><family>Boss</family> <given>Big</given></name>
   <email>chief@foo.com</email>
   <link subordinates="one.worker two.worker three.worker four.worker
     five.worker"/>
 </person>

 <person id="one.worker">
   <name><family>Worker</family> <given>One</given></name>
   <email>one@foo.com</email>
   <link manager="Big.Boss"/>
 </person>
  ......

</personnel>
```

DTDs

- A DTD describes the structure of the XML document
- It defines:
  - Each element (tag) that is permitted
  - The attributes allowed for each element
  - The content model of each element

    May specify nested elements, either as a sequence or choice.

    Or may be parsed character data (#PCDATA)

DTD Example

```
<?xml encoding="ISO-8859-1"?>

<!ELEMENT personnel (person)+>
<!ELEMENT person (name,email*,url*,link?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT name (#PCDATA|family|given)*>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url EMPTY>
<!ATTLIST url href CDATA #REQUIRED>
<!ELEMENT link EMPTY>
<!ATTLIST link
  manager IDREF #IMPLIED
  subordinates IDREFS #IMPLIED>
```

Schemas

- Also describe the structure of the XML document, but in much more detail
  - Allow constraints on elements and attributes (i.e., date, numeric, list of values, mask, etc.)
  - Min/max repeat counts
- Other functions and constructs that are not supported in DTDs
  - “all” content spec
  - Ability to define your own types, including base and derived types.

Schema Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="personnel">
      <xs:complexType>
          <xs:sequence>
                <xs:element ref="person" minOccurs="1" maxOccurs="unbounded" />
          </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="person">
      <xs:complexType>
          <xs:sequence>
                <xs:element ref="name" />
                <xs:element name ="email" minOccurs="0" maxOccurs="3" type="xs:string" />
                <xs:element ref="url" minOccurs="0" maxOccurs="unbounded" />
                <xs:element ref="link" minOccurs="0" maxOccurs="1" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required" />
      <xs:complexType>
</xs:element>
….
</xs:schema>
```

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/ns/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/ns/POExample:Address>

Namespace Example

```
<po:OrderSR-S
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:po="http://www.example.com/PO1"
  xsi:schemaLocation="http://www.example.com/PO1 poxml5sr-schema.xsd">
   <po:Header typecode="00">
     <po:PONum>PO12345678901234</po:PONum>
     ……..
```

Converting between EDI and XML - Overview

- Setup is the same for all platforms
  - Uses WDI Client GUI interface
  - Defines how and when to do the transformation
  - Can be done in local database and exported to server database (client standalone mode)
  - Or, can be done directly in server database (client-server mode)
- Runtime execution varies by platform

Converting between EDI and XML - Setup

- Import XML DTD or schema
  - Obtained from trading partner or industry group
  - Or, created by user
- Import EDI standard
  - Obtained from WDI web site
- Create the *map* using WDI Client
  - Defines how the data is converted
- Create a *rule* for the map
  - Defines when this map is used, additional options

Special mapping commands and properties

- DIProlog property
  - Allows you to specify a custom prolog (<?xml…)
- SetNoNSSchemaLocation("location")
  - Creates: xsi:noNamespaceSchemaLocation attribute
- SetSchemaLocation(URI)
  - Creates: xsi:schemaLocation attribute
- SetNamespace(URI)
  - Creates: xmlns:prefix attribute

Converting between EDI and XML - Runtime

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

PERFORM TRANSFORM example

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

Special XML keywords for TRANSFORM

- XMLDTDS – Directory or PDS containing DTDs and schemas
- XMLEBCDIC(Y/N) – Force input to be interpreted as EBCDIC
  - z/OS only
- XMLNS(Y/N) – Namespace processing for input
  - Always use Y if dealing with XML schema input
- XMLSCHEMAVAL(Y/N/A) – Do schema validation
  - Y and A also override XMLVALIDATE value
  - Not available on CICS TS 1.3
- XMLVALIDATE(0/1/2) – Controls DTD validation and use
- XMLSPLIT(Y/N) – "Deenveloping" for XML (Coming soon!)

Demo

- Mapping and transformation demo
  - XML (based on DTD) to EDI
  - EDI to XML (based on schema)

Summary

- XML is becoming a very common method of exchanging data.
- Define the structure of the XML document to WDI by importing the DTDs or schemas.
- Map the XML document using Data Transformation maps.
- Translate the data using PERFORM TRANSFORM command.

Questions ??

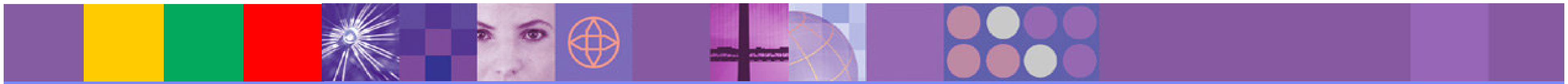
Any Questions ??

Follow-up

- Additional questions
  - Fritz Fahrenback (fritzf@us.ibm.com)
  - Lynn Clark (emd19@us.ibm.com)
- WebSphere Data Interchange
  - http://www.ibm.com/websphere/datainterchange

Thank you for attending !

WebSphere. software



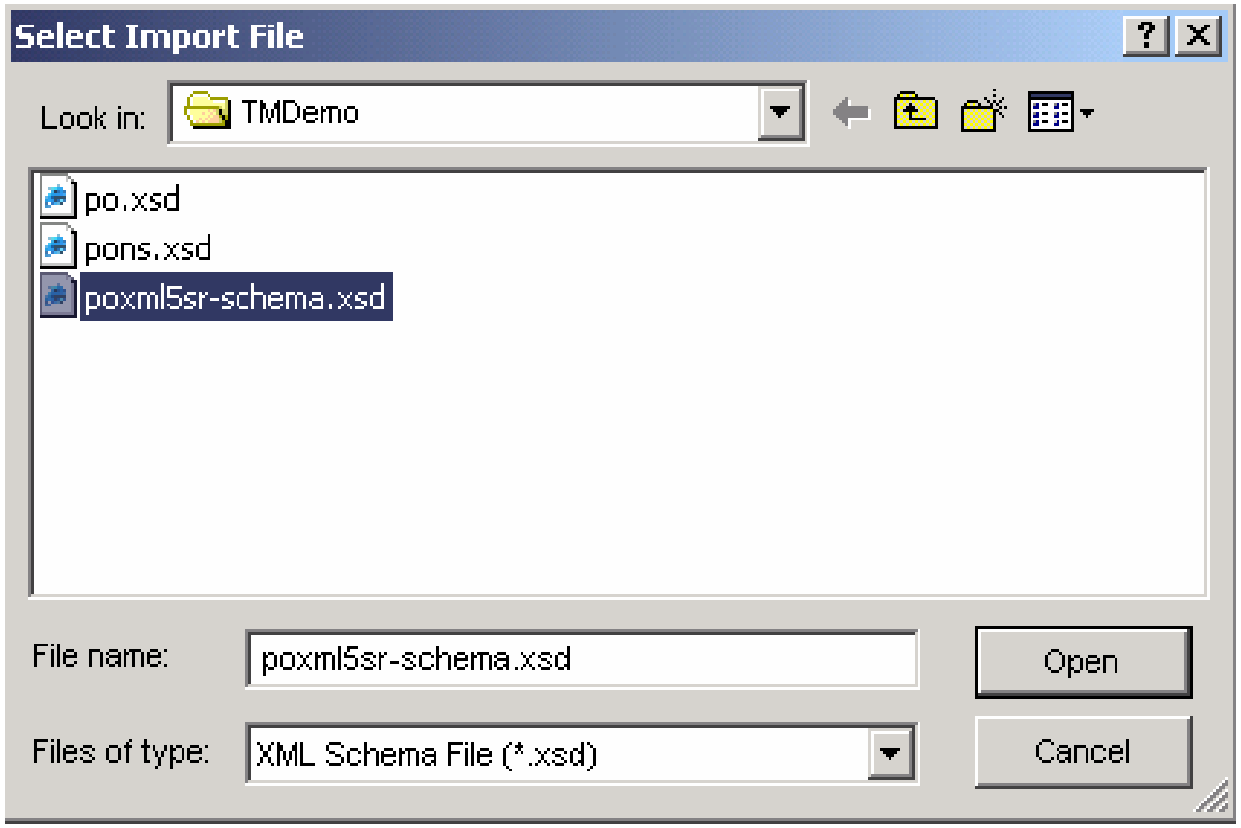
business software

Screen shots from demo

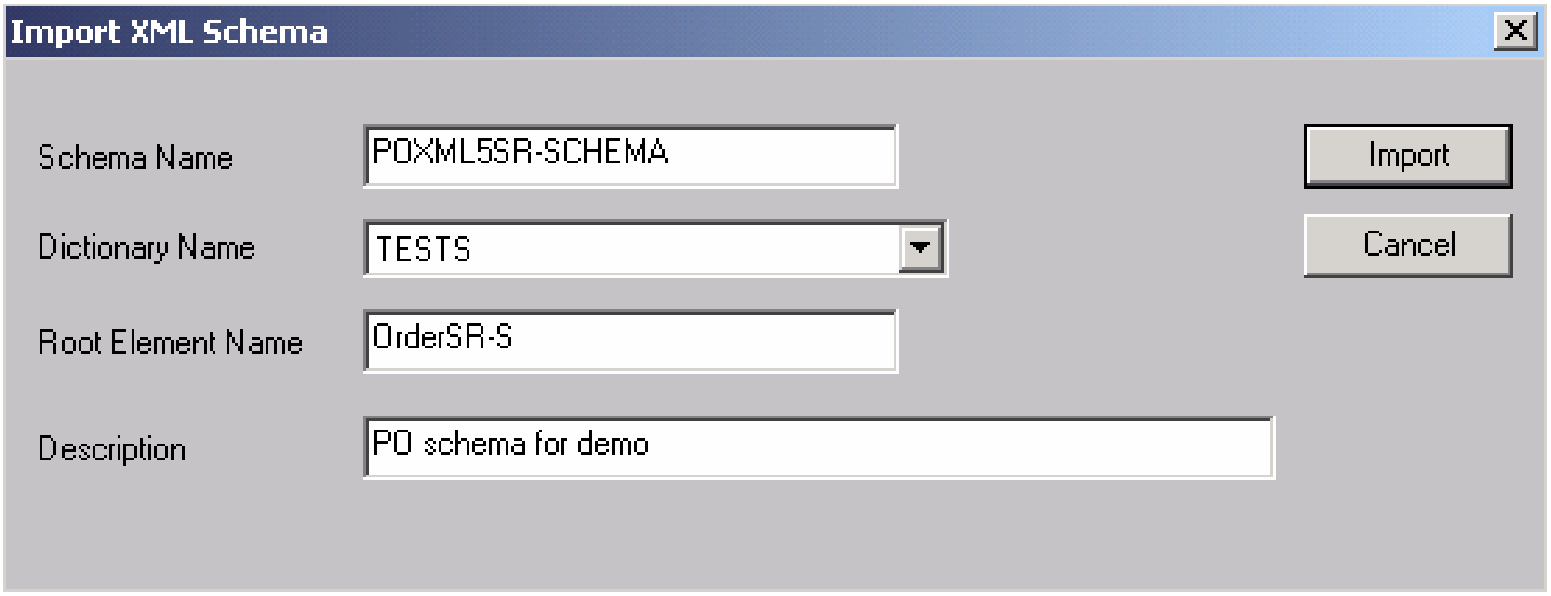
WebSphere. software

e business software

Importing a DTD or schema

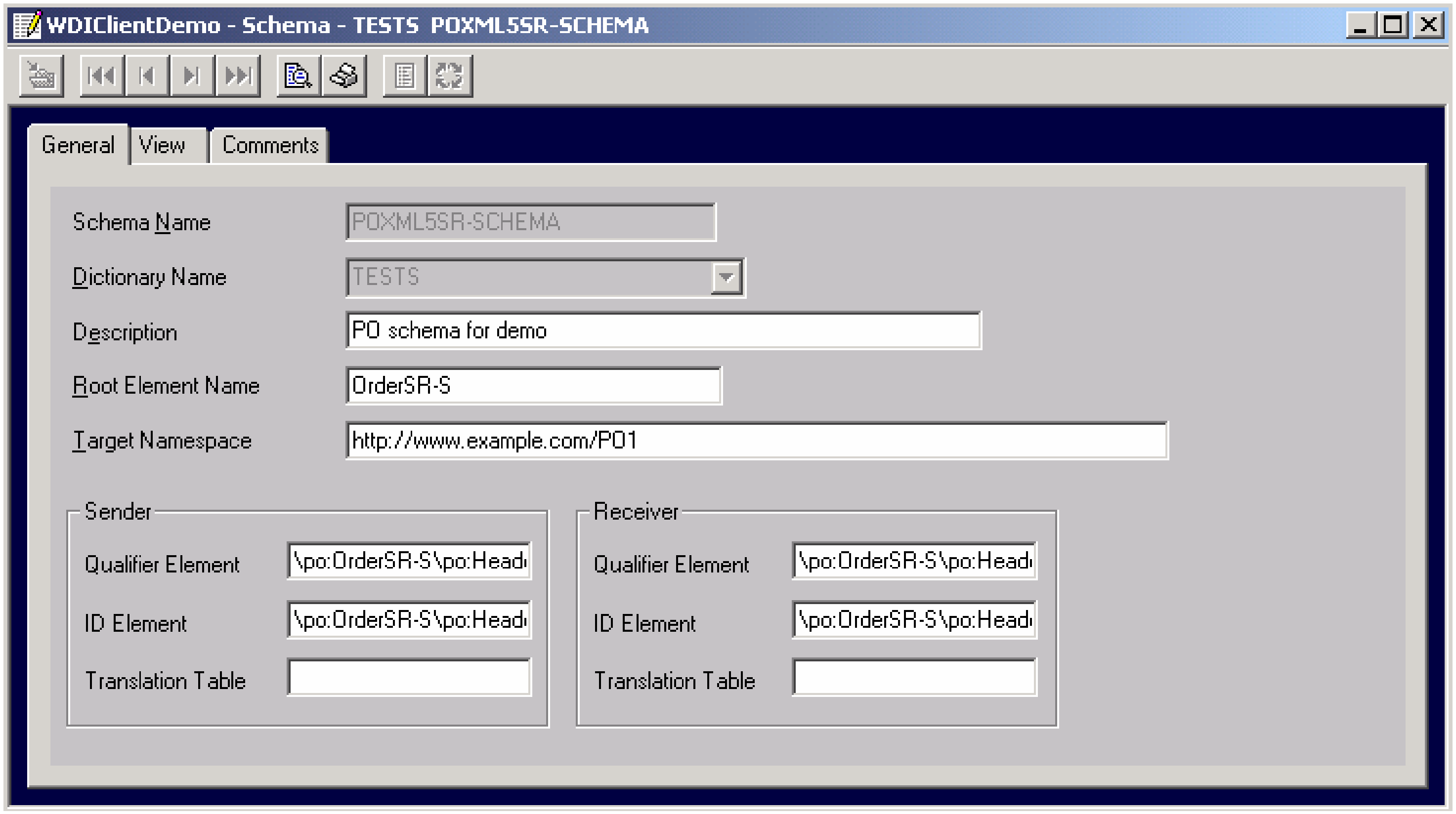


Importing a DTD or schema



Schema Name	POXML5SR-SCHEMA
Dictionary Name	TESTS
Root Element Name	OrderSR-S
Description	PO schema for demo

Defining Sender and Receiver elements


WDIClientDemo - Schema - TESTS POXML5SR-SCHEMA

Schema Name	POXML5SR-SCHEMA
Dictionary Name	TESTS
Description	PO schema for demo
Root Element Name	OrderSR-S
Target Namespace	http://www.example.com/PO1

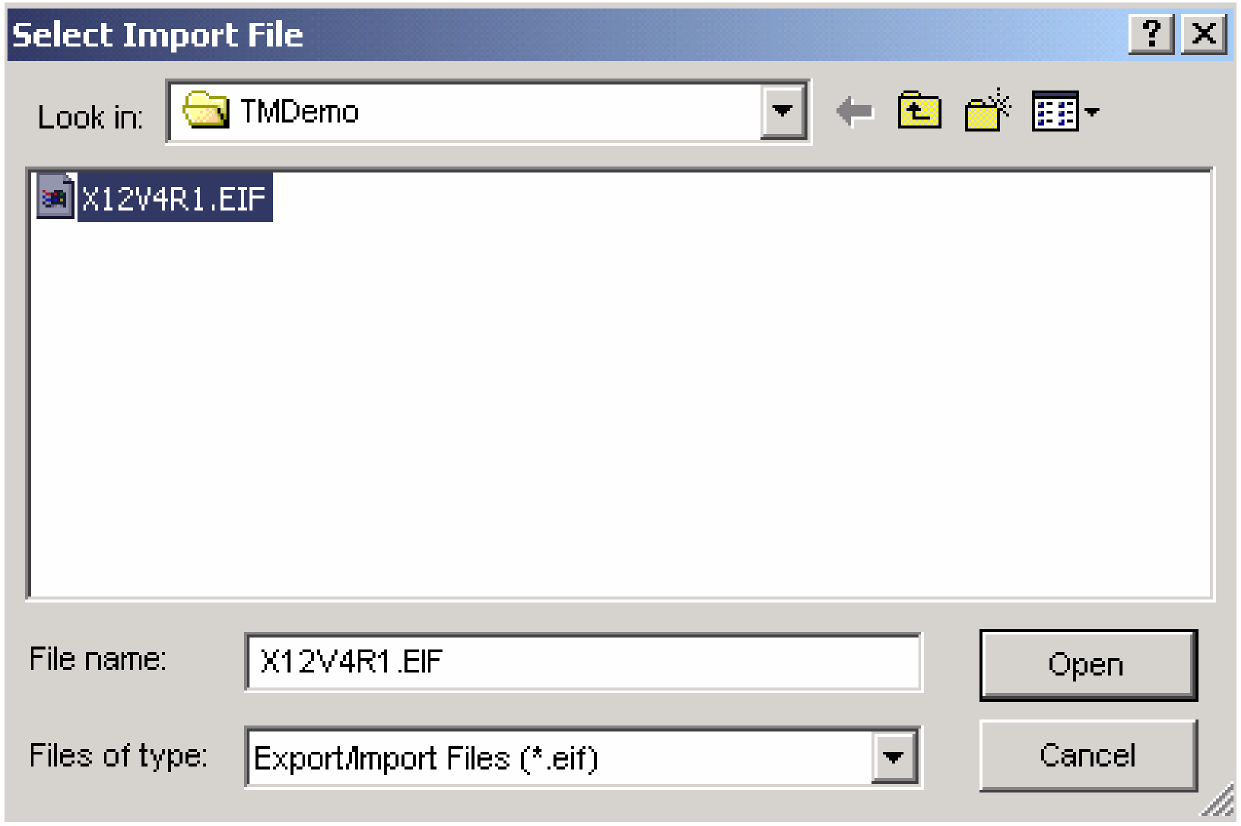
Sender

Qualifier Element	\po:OrderSR-S\po:Head
ID Element	\po:OrderSR-S\po:Head
Translation Table	

Receiver

Qualifier Element	\po:OrderSR-S\po:Head
ID Element	\po:OrderSR-S\po:Head
Translation Table	

Importing EDI Standard



Select Import File

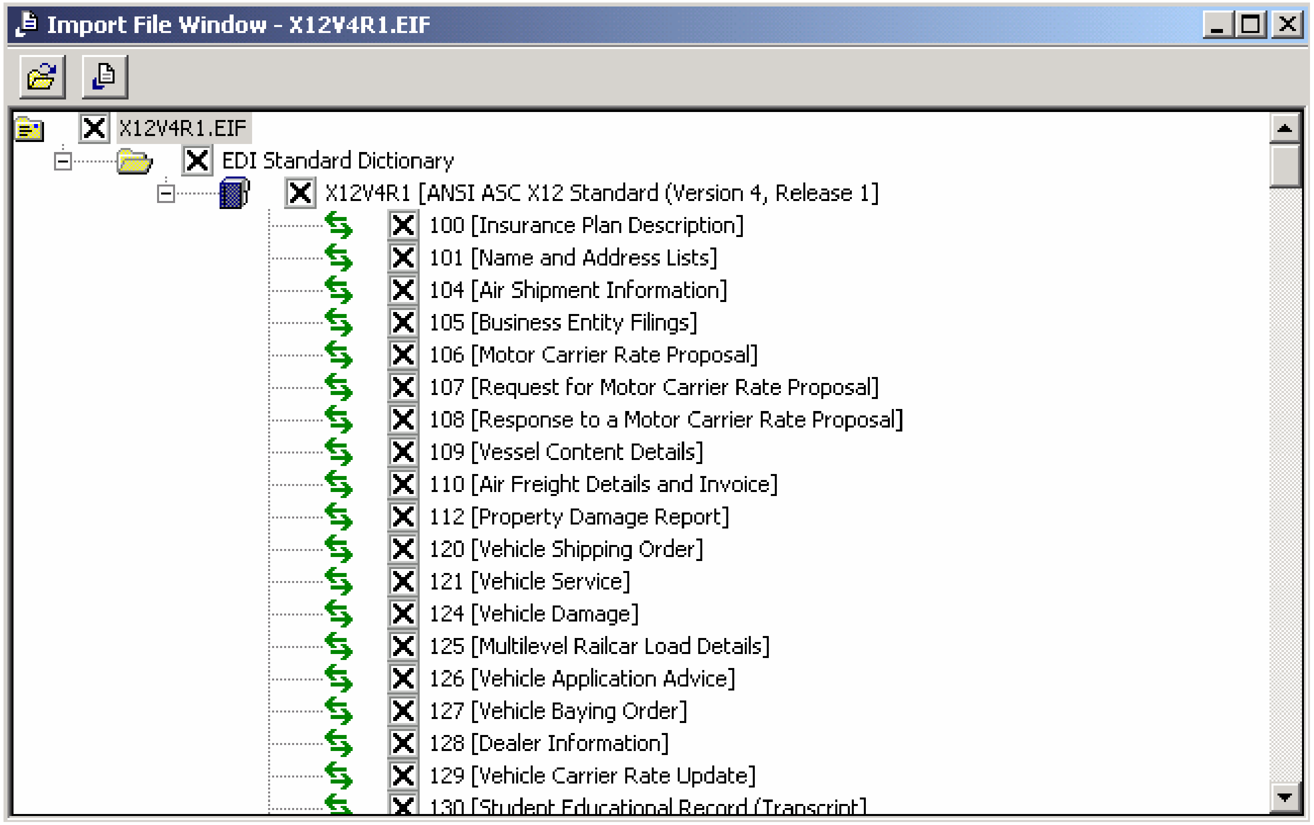
Look in: TMDemo

X12V4R1.EIF

File name: X12V4R1.EIF

Files of type: Export/Import Files (*.eif)

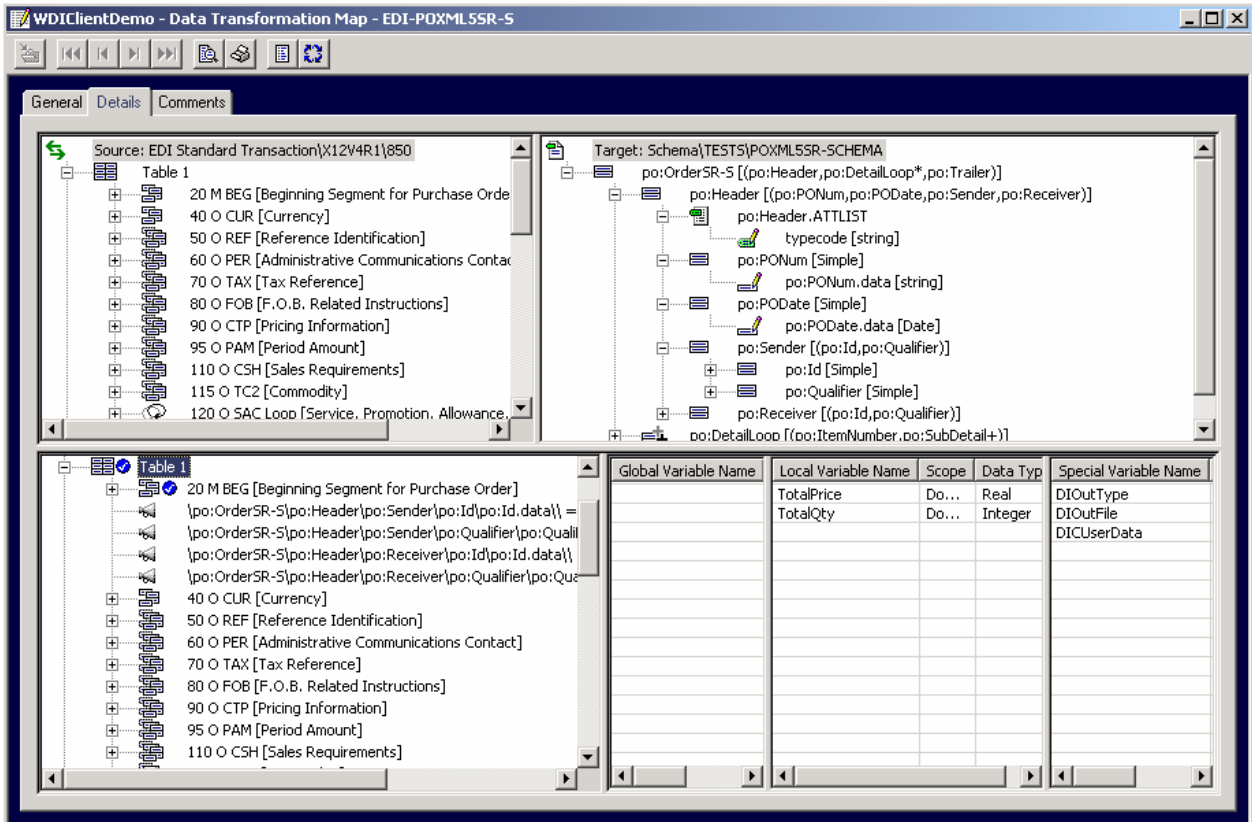
Importing EDI Standard



Import File Window - X12V4R1.EIF

```
X12V4R1.EIF
  EDI Standard Dictionary
    X12V4R1 [ANSI ASC X12 Standard (Version 4, Release 1]
      100 [Insurance Plan Description]
      101 [Name and Address Lists]
      104 [Air Shipment Information]
      105 [Business Entity Filings]
      106 [Motor Carrier Rate Proposal]
      107 [Request for Motor Carrier Rate Proposal]
      108 [Response to a Motor Carrier Rate Proposal]
      109 [Vessel Content Details]
      110 [Air Freight Details and Invoice]
      112 [Property Damage Report]
      120 [Vehicle Shipping Order]
      121 [Vehicle Service]
      124 [Vehicle Damage]
      125 [Multilevel Railcar Load Details]
      126 [Vehicle Application Advice]
      127 [Vehicle Baying Order]
      128 [Dealer Information]
      129 [Vehicle Carrier Rate Update]
      130 [Student Educational Record (Transcript]
```

EDI to XML map



WDIClientDemo - Data Transformation Map - EDI-POXML5SR-S

General | Details | Comments

Source: EDI Standard Transaction\X12V4R1\850
Table 1
20 M BEG [Beginning Segment for Purchase Orde
40 O CUR [Currency]
50 O REF [Reference Identification]
60 O PER [Administrative Communications Contac
70 O TAX [Tax Reference]
80 O FOB [F.O.B. Related Instructions]
90 O CTP [Pricing Information]
95 O PAM [Period Amount]
110 O CSH [Sales Requirements]
115 O TC2 [Commodity]
120 O SAC Loop [Service, Promotion, Allowance,

Target: Schema\TESTS\POXML5SR-SCHEMA
po:OrderSR-S [(po:Header,po:DetailLoop*,po:Trailer)]
po:Header [(po:PONum,po:PODate,po:Sender,po:Receiver)]
po:Header.ATTLIST
typecode [string]
po:PONum [Simple]
po:PONum.data [string]
po:PODate [Simple]
po:PODate.data [Date]
po:Sender [(po:Id,po:Qualifier)]
po:Id [Simple]
po:Qualifier [Simple]
po:Receiver [(po:Id,po:Qualifier)]
po:DetailLoop [(po:ItemNumber,po:SubDetail+)]

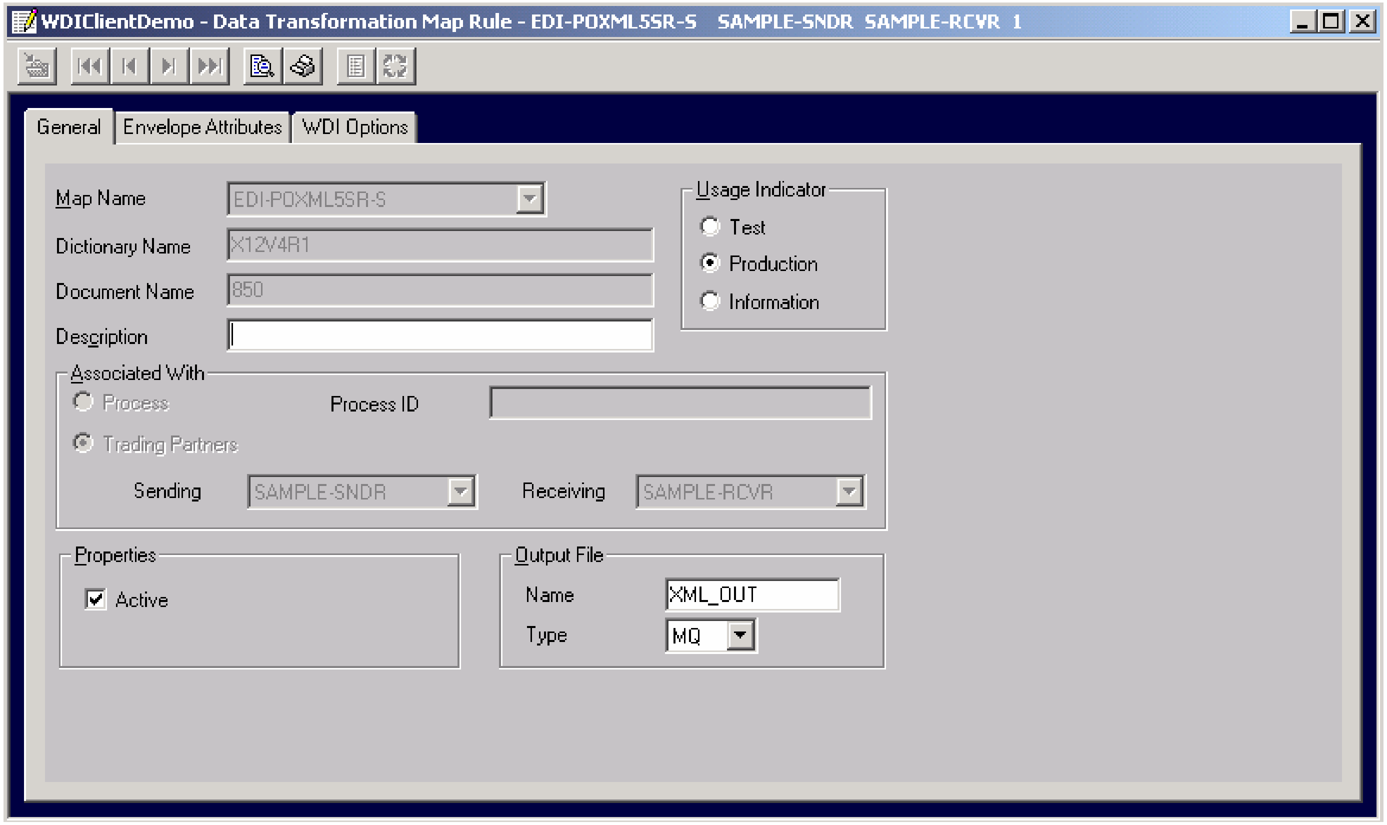
Table 1
20 M BEG [Beginning Segment for Purchase Order]
\po:OrderSR-S\po:Header\po:Sender\po:Id\po:Id.data\\ =
\po:OrderSR-S\po:Header\po:Sender\po:Qualifier\po:Quali
\po:OrderSR-S\po:Header\po:Receiver\po:Id\po:Id.data\\
\po:OrderSR-S\po:Header\po:Receiver\po:Qualifier\po:Qua
40 O CUR [Currency]
50 O REF [Reference Identification]
60 O PER [Administrative Communications Contact]
70 O TAX [Tax Reference]
80 O FOB [F.O.B. Related Instructions]
90 O CTP [Pricing Information]
95 O PAM [Period Amount]
110 O CSH [Sales Requirements]

Global Variable Name

Local Variable Name	Scope	Data Typ
TotalPrice	Do...	Real
TotalQty	Do...	Integer

Special Variable Name
DIOutType
DIOutFile
DICUserData

Creating a Rule


WDIClientDemo - Data Transformation Map Rule - EDI-POXML5SR-S SAMPLE-SNDR SAMPLE-RCVR 1

General | Envelope Attributes | WDI Options

Map Name: EDI-POXML5SR-S

Dictionary Name: X12V4R1

Document Name: 850

Description:

Usage Indicator: Test [✓ Production] Information

Associated With: Process [✓ Trading Partners]

Process ID:

Sending: SAMPLE-SNDR

Receiving: SAMPLE-RCVR

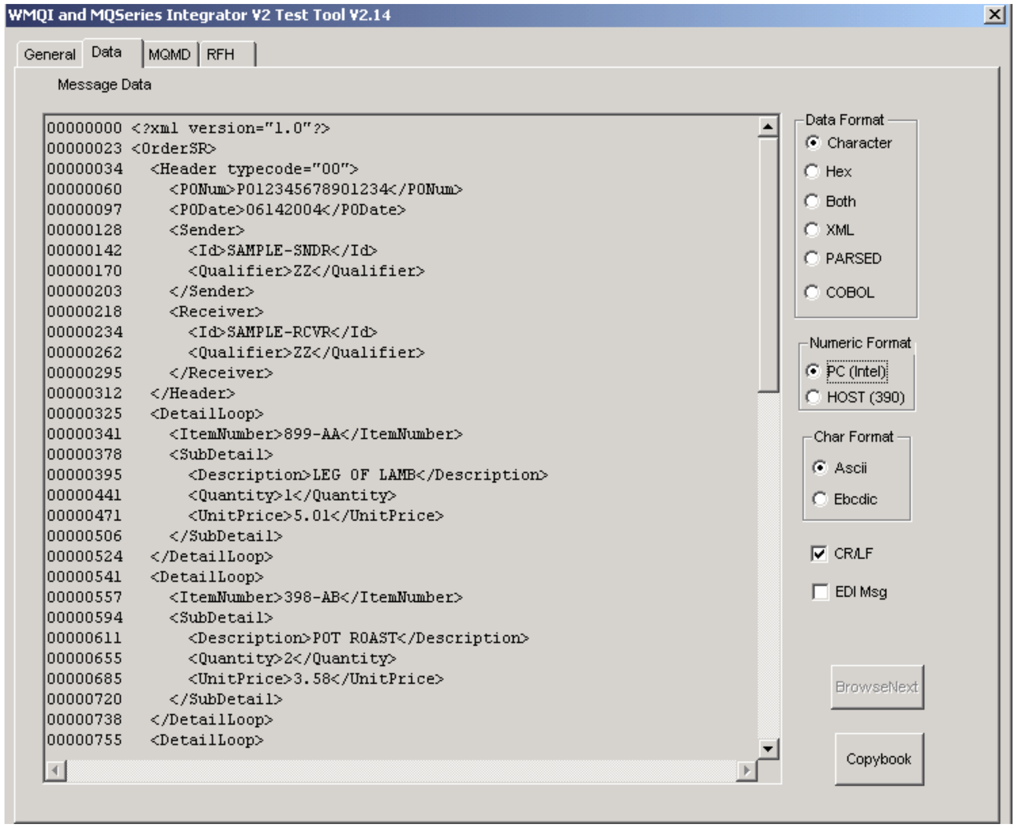
Properties: [✓ Active]

Output File

Name: XML_OUT

Type: MQ

XML Input
(XML-EDI translation)



```
00000000 <?xml version="1.0"?>
00000023 <OrderSR>
00000034   <Header typecode="00">
00000060     <PONum>P012345678901234</PONum>
00000097     <PODate>06142004</PODate>
00000128     <Sender>
00000142       <Id>SAMPLE-SNDR</Id>
00000170       <Qualifier>ZZ</Qualifier>
00000203     </Sender>
00000218     <Receiver>
00000234       <Id>SAMPLE-RCVR</Id>
00000262       <Qualifier>ZZ</Qualifier>
00000295     </Receiver>
00000312   </Header>
00000325   <DetailLoop>
00000341     <ItemNumber>899-AA</ItemNumber>
00000378     <SubDetail>
00000395       <Description>LEG OF LAMB</Description>
00000441       <Quantity>1</Quantity>
00000471       <UnitPrice>5.01</UnitPrice>
00000506     </SubDetail>
00000524   </DetailLoop>
00000541   <DetailLoop>
00000557     <ItemNumber>398-AB</ItemNumber>
00000594     <SubDetail>
00000611       <Description>POT ROAST</Description>
00000655       <Quantity>2</Quantity>
00000685       <UnitPrice>3.58</UnitPrice>
00000720     </SubDetail>
00000738   </DetailLoop>
00000755   <DetailLoop>
```

Data Format: [✓ Character] Hex Both XML PARSED COBOL

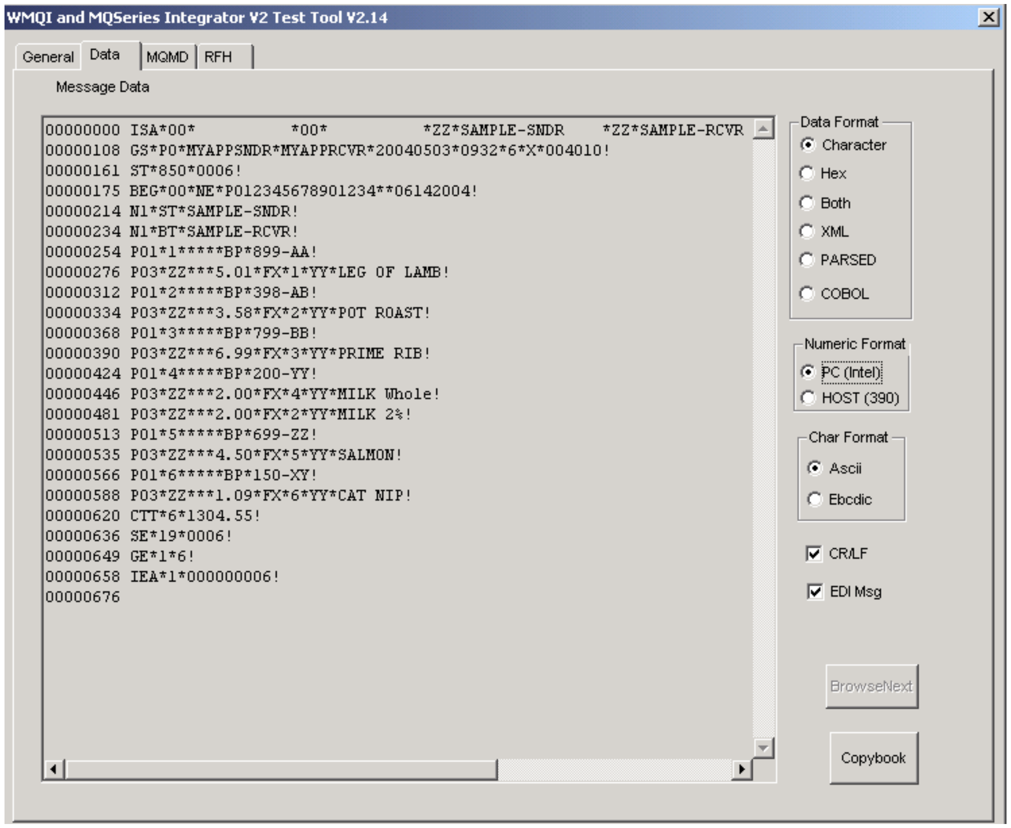
Numeric Format: [✓ PC (Intel)] HOST (390)

Char Format: [✓ Ascii] Ebcdic

[✓ CR/LF]

EDI Msg

EDI Output (XML-EDI)
EDI Input (EDI-XML)



Data Format: [✓ Character] Hex Both XML PARSED COBOL

Numeric Format: [✓ PC (Intel)] HOST (390)

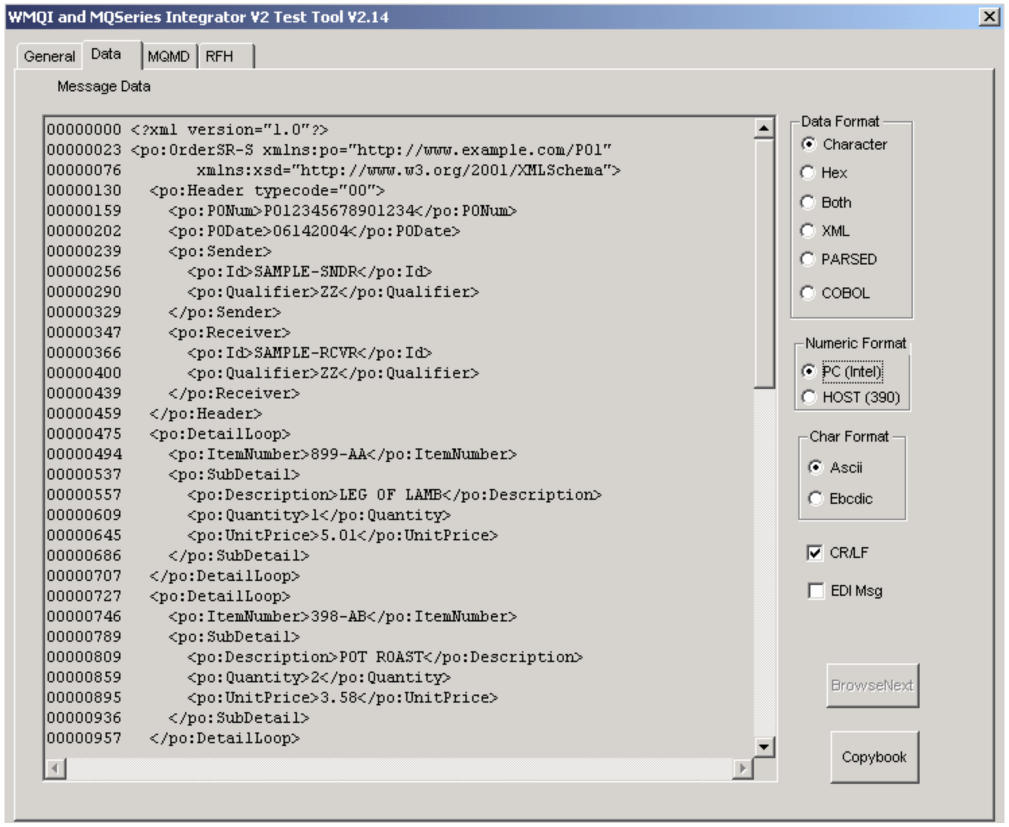
Char Format: [✓ Ascii] Ebcdic

[✓ CR/LF]

[✓ EDI Msg]

```
00000000 ISA*00*          *00*          *ZZ*SAMPLE-SNDR    *ZZ*SAMPLE-RCVR
00000108 GS*PO*MYAPPSNDR*MYAPPRCVR*20040503*0932*6*X*004010!
00000161 ST*850*0006!
00000175 BEG*00*NE*P012345678901234**06142004!
00000214 N1*ST*SAMPLE-SNDR!
00000234 N1*BT*SAMPLE-RCVR!
00000254 PO1*1*****BP*899-AA!
00000276 PO3*ZZ***5.01*FX*1*YY*LEG OF LAMB!
00000312 PO1*2*****BP*398-AB!
00000334 PO3*ZZ***3.58*FX*2*YY*POT ROAST!
00000368 PO1*3*****BP*799-BB!
00000390 PO3*ZZ***6.99*FX*3*YY*PRIME RIB!
00000424 PO1*4*****BP*200-YY!
00000446 PO3*ZZ***2.00*FX*4*YY*MILK Whole!
00000481 PO3*ZZ***2.00*FX*2*YY*MILK 2%!
00000513 PO1*5*****BP*699-ZZ!
00000535 PO3*ZZ***4.50*FX*5*YY*SALMON!
00000566 PO1*6*****BP*150-XY!
00000588 PO3*ZZ***1.09*FX*6*YY*CAT NIP!
00000620 CTT*6*1304.55!
00000636 SE*19*0006!
00000649 GE*1*6!
00000658 IEA*1*000000006!
00000676
```

XML Output (EDI-XML translation)



```
00000000 <?xml version="1.0"?>
00000023 <po:OrderSR-S xmlns:po="http://www.example.com/PO1"
00000076        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
00000130   <po:Header typecode="00">
00000159     <po:PONum>PO12345678901234</po:PONum>
00000202     <po:PODate>06142004</po:PODate>
00000239     <po:Sender>
00000256       <po:Id>SAMPLE-SNDR</po:Id>
00000290       <po:Qualifier>ZZ</po:Qualifier>
00000329     </po:Sender>
00000347     <po:Receiver>
00000366       <po:Id>SAMPLE-RCVR</po:Id>
00000400       <po:Qualifier>ZZ</po:Qualifier>
00000439     </po:Receiver>
00000459   </po:Header>
00000475   <po:DetailLoop>
00000494     <po:ItemNumber>899-AA</po:ItemNumber>
00000537     <po:SubDetail>
00000557       <po:Description>LEG OF LAMB</po:Description>
00000609       <po:Quantity>1</po:Quantity>
00000645       <po:UnitPrice>5.01</po:UnitPrice>
00000686     </po:SubDetail>
00000707   </po:DetailLoop>
00000727   <po:DetailLoop>
00000746     <po:ItemNumber>398-AB</po:ItemNumber>
00000789     <po:SubDetail>
00000809       <po:Description>POT ROAST</po:Description>
00000859       <po:Quantity>2</po:Quantity>
00000895       <po:UnitPrice>3.58</po:UnitPrice>
00000936     </po:SubDetail>
00000957   </po:DetailLoop>
```

Data Format: [✓ Character] Hex Both XML PARSED COBOL

Numeric Format: [✓ PC (Intel)] HOST (390)

Char Format: [✓ Ascii] Ebcdic

[✓ CR/LF]

☐ EDI Msg