



XML, COBOL and Application Modernization

Bringing XML to the application

Tom Ross
Nick Tindall
Share Tampa
February, 2007



© IBM 2007



Agenda

- **Some XML terminology**
- **Scenarios**
- **XML support in COBOL**
- **WebSphere Developer – XSE**
 - XML Services for the Enterprise
- **Discussion, questions**





e-business

Agenda...

- **Some XML terminology**



Some XML terminology

- **XML processing programs (“parsers”):**
 - SAX - Simple API for XML
 - ▶ Callbacks to your program for each XML “event” (document fragment)
 - DOM - Document Object Model
 - ▶ In effect the XML document becomes a miniature data base that supports navigation and simple queries
- **Well-formed XML document - basic grammar rules:**
 - exactly one “root” element
 - a matching end tag for every start tag
 - elements properly nested, etc., etc.
- **Valid XML document (wrt a DTD or schema):**
 - Well-formedness, plus...
 - Element order and containment (structure constraints)
 - Proper content (type constraints)



e-business

Agenda...

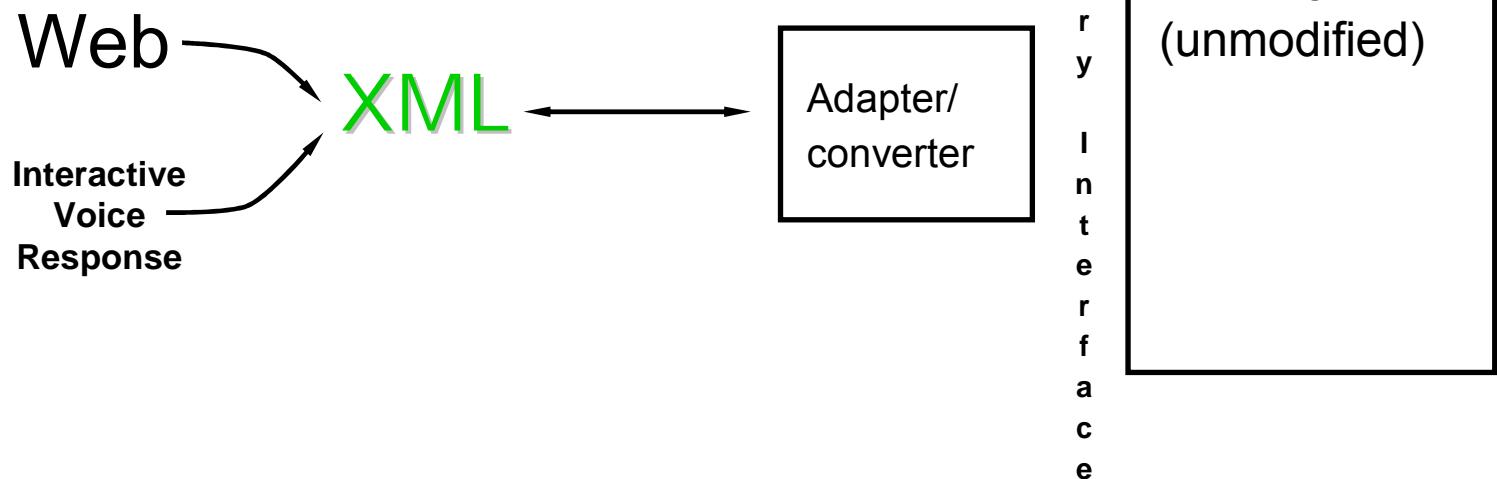
- Some XML terminology
- **Scenarios**



Customer scenarios/problems - 1

■ Refacing existing applications/transactions

- To allow non-traditional clients to use them
- A step towards componentization
- In particular to allow access as a Web Service
- Various transports/protocols
- Performance typically critical





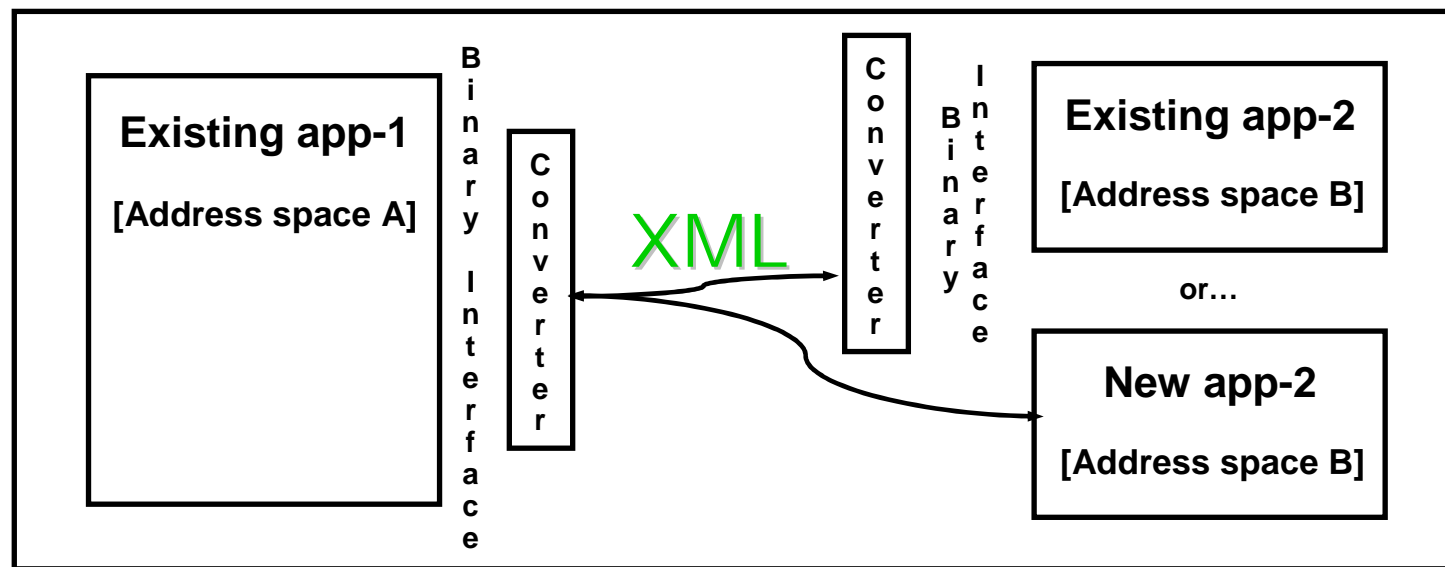
e-business

Customer scenarios/problems - 2

■ Inter-application communication

- Need arises because of mergers/acquisitions, or
- Between different purchased applications
- Interfaces need only be compatible, not identical
- Allows schema evolution without synchronized updates
- Would previously have been done with file readers/writers

Single Enterprise system/complex



© IBM 2007



e-business

Agenda...

- Some XML terminology
- Scenarios
- **XML support in COBOL**





Why process XML in applications?

- **Coherent development context and methodology**
- **Centralizes business logic within the application**
 - Versus some business function here, some there, ...
- **Independent of middleware choices, characteristics**
- **Allows business logic to be conveniently applied during and after message acquisition/generation**
- **Incremental step from existing application design**
- **Can process XML messages as such**
 - Versus forcing conversion to traditional data structures





Why process XML in COBOL?

- **Keeps development control in one place/style**
- **Guarantees correct language semantics**
 - sign configuration
 - layout/padding
 - picture constraints
- **Exploits your existing assets/skills/literacy**
- **Supports a variety of scenarios:**
 - Converters, bridging to existing (unchanged) applications
 - Direct use of XML in new or enhanced applications
- **High performance: CPU and elapsed time**





Enterprise COBOL XML support

- **Inbound XML parsing and outbound XML generation:**
 - XML PARSE xml-doc PROCESSING PROCEDURE xml-handler
 - XML GENERATE xml-doc FROM data-item COUNT IN char-count
- **XML parsing**
 - Much faster than general purpose parsers
 - ▶ Designed for high-speed transaction processing
 - Runs in all COBOL run-time environments:
 - ▶ CICS, IMS, batch, TSO, USS, ...
 - Works with any transport mechanism for XML documents
 - ▶ Use MQSeries, CICS transient queue or COMMAREA, IMS message processing queue, WebSphere, etc.
 - Provides basic SAX-style parsing
 - Checks well-formedness (but not validity)
 - XML Parser is part of the run-time library
 - ▶ Can be used from Enterprise COBOL or Enterprise PL/I
- **XML generation:**
 - Single statement transforms entire data structure (group)





e-business

XML PARSE

- **Parses XML documents that are in memory, in a COBOL alphanumeric or national data item**
- **Discovers the individual pieces of XML documents**
 - Passes each piece to user-written processing procedure
- **During parsing you can populate COBOL data structures with the data from XML messages**
 - Advantage: non-COBOL programs can communicate data to/from COBOL without having to know the COBOL data structure formats!



IBM

© IBM 2007



e-business

XML PARSE...

■ XML PARSE statement

- The COBOL interface to the high-speed XML parser

■ XML special registers

- XML-CODE: communicates status of parsing
- XML-EVENT: describes each event during parsing
- XML-TEXT: contains XML document fragments
- XML-NTEXT: contains NATIONAL XML doc fragments

```
XML PARSE XMLDOCUMENT
```

```
    PROCESSING PROCEDURE XMLEVENT-HANDLER
```

```
END-XML
```

```
...
```

```
XMLEVENT-HANDLER.
```

```
    EVALUATE TRUE
```

```
        WHEN XML-EVENT = 'START-OF-ELEMENT' AND
```

```
            XML-TEXT = 'TRADE'
```

```
                DISPLAY 'Processing new stock trade'
```

```
...
```

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font with horizontal stripes.

© IBM 2007



Hello XML World — inbound

Identification division.

Program-id. HelloXML.

Data division.

Working-storage section.

1 M.

2 pic x(21) value '<?xml version="1.0"?>'.
2 pic x(40) value '<msg type="succinct">Hello, World!</msg>'.
Procedure division.

Display 'XML Event XML Text'

XML Parse M

Processing procedure P

End-XML

Goback.

P.

If XML-Code = 0

Display XML-Event XML-Text

End-if.

End program HelloXML.

| XML Event | XML Text |
|----------------------|---|
| START-OF-DOCUMENT | <?xml version="1.0"?><msg type="succinct">Hello, World!</msg> |
| VERSION-INFORMATION | 1.0 |
| START-OF-ELEMENT | msg |
| ATTRIBUTE-NAME | type |
| ATTRIBUTE-CHARACTERS | succinct |
| CONTENT-CHARACTERS | Hello, World! |
| END-OF-ELEMENT | msg |
| END-OF-DOCUMENT | |





XML GENERATE ... FROM

- **Generates a complete XML document:**
 - Into an alphanumeric or national data item
 - From a group or elementary item
- **Data values are “trimmed:”**
 - Trailing spaces removed from alphanumeric values (leading spaces from right-justified items)
 - Leading zeroes removed from numeric values
- **XML tag names are the mixed-case data names**
- **Supports all data types except -pointer and object reference**
- **Unnamed and redefining items are ignored**
- **For more details see V3.4 Programming Guide:**
 - www.ibm.com/software/awdtools/cobol/zos/library/





e-business

Hello XML World — outbound

Identification division.

Program-id. HelloXML.

Data division.

Working-storage section.

1 Hello-doc pic x(200).

1 **Greeting**.

2 **msg** pic x(100) value '**Hello, World!** '.

1 Num-chars pic 999.

Procedure division.

XML Generate Hello-doc From Greeting
Count in Num-chars

Display '| ' Hello-doc(1:Num-chars) '| '
Goback.

End program HelloXML.

```
|<Greeting><msg>Hello, World!</msg></Greeting>|
```

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font with horizontal stripes.

© IBM 2007



e-business

Hello XML World — outbound ...

```
1 Hello-doc pic x(200).
1 Greeting-Grp.
  2 Contact-info.
    3 Name          PIC x(20) Value 'Tom'.
    3 Addr          PIC x(18) Value '555 Bailey Ave'.
    3 Telephone PIC 9(12) Value 4084634242.
  2 Contact-redef REDEFINES Contact-info.
    3 Junk PIC X(50).
  2 msg pic x(100) value 'Hello, World! '.
1 Greeting-redef REDEFINES Greeting-Grp.
  2 Name          PIC x(20).
  2              PIC x(18).
  2 Phone        PIC 9(12).
  2 Short-Msg    PIC X(10).
1 Num-chars binary pic 9(9).
```



© IBM 2007



e-business

Hello XML World — outbound ...

XML Generate Hello-doc From [Greeting-grp](#) Count in Num-chars

```
<Greeting-grp>
  <Contact-info>
    <Name>Tom</Name>
    <Addr>555 Bailey Ave</Addr>
    <Telephone>4084634242</Telephone>
    <msg>Hello, World!</msg>
  </Contact-info>
</Greeting-grp>
```

XML Generate Hello-doc From [Greeting-redef](#) Count in Num-chars

```
<Greeting-redef>
  <Name>Tom</Name>
  <Phone>4084634242</Phone>
  <Short-Msg>Hello, World!</Short-Msg>
</Greeting-redef>
```

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font, with horizontal lines through the letters.

© IBM 2007



Agenda...

- Some XML terminology
- Scenarios
- XML support in COBOL
- **WebSphere Developer for zSeries (WD for z)**
XML Services for the Enterprise (XSE)





e-business

What is WD XSE?

- **Allows COBOL applications to consume and produce XML messages**
- **Leverages XML parsing capabilities of IBM Enterprise COBOL V3**
- **Creates:**
 - *Inbound* converter program, to convert XML messages into native COBOL data
 - *Outbound* converter program, to convert native COBOL data into XML messages
 - Sample COBOL driver program:
 - ▶ Illustrates the invocation of converters
 - ▶ Illustrates the interaction with existing application
 - ▶ Needs to be modified before use
 - XML Schema, for validation and generating samples
 - ▶ Input to the Web Service Description (WSDL)
- **Enables communication with XML-based systems**

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font, with horizontal lines through the letters.

© IBM 2007



e-business



© IBM 2007

WD XSE

- **Web Service Runtime and Scenario Selection** dialog
- **Create New Service Interface (bottom-up)** wizard
 - **Compiled XML conversion type**
 - **Language structures page**
 - **Generation options page**
 - **BIDI conversion options dialog**
 - **Runtime specific pages**
 - **Web Services for CICS page**
 - **IMS SOAP Access page**
 - **File, data set or member selection**
 - **Interpretive XML conversion type (bottom-up)**
 - **CICS Web Services Assistant pages (bottom-up)**
- **Create New Service Implementation (top-down)** wizard
 - **Interpretive XML conversion type (top-down)**
 - **CICS Web Services Assistant pages (top-down)**
- **Map to an Existing Service Interface (meet-in-the-middle)**
 - **Mapping Converter Generator** wizard
- **Generating artifacts remotely**



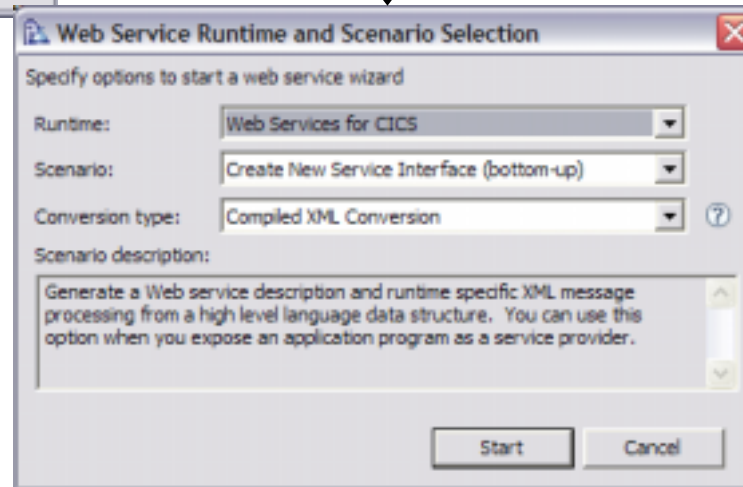
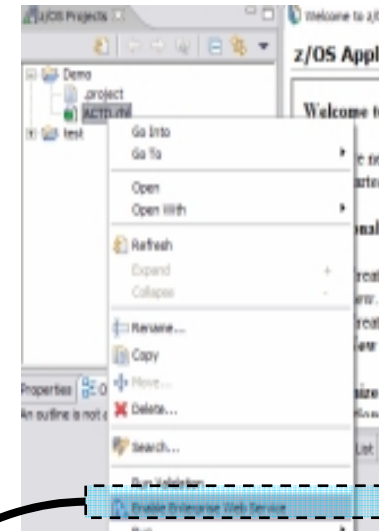
e-business

WD XSE

Web Service Runtime and Scenario Selection dialog

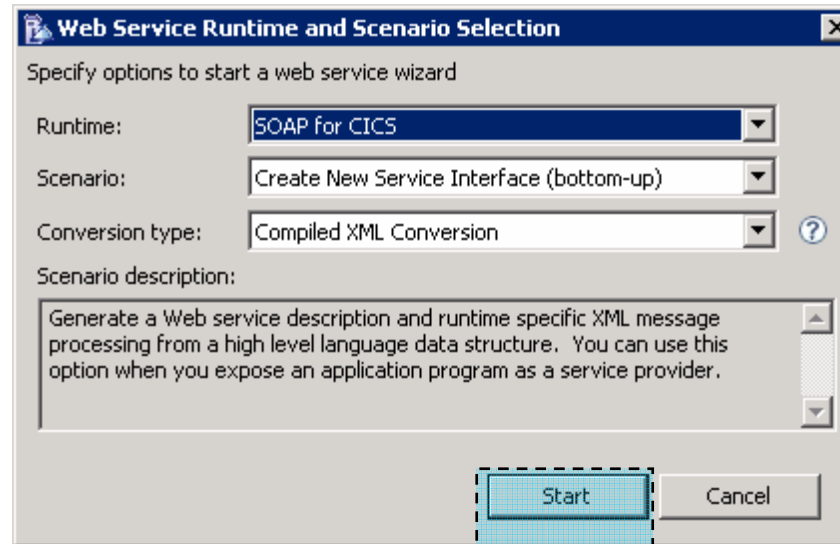
Select artifact then File->New->Other...

OR Enable Enterprise Web service from pop-up menu



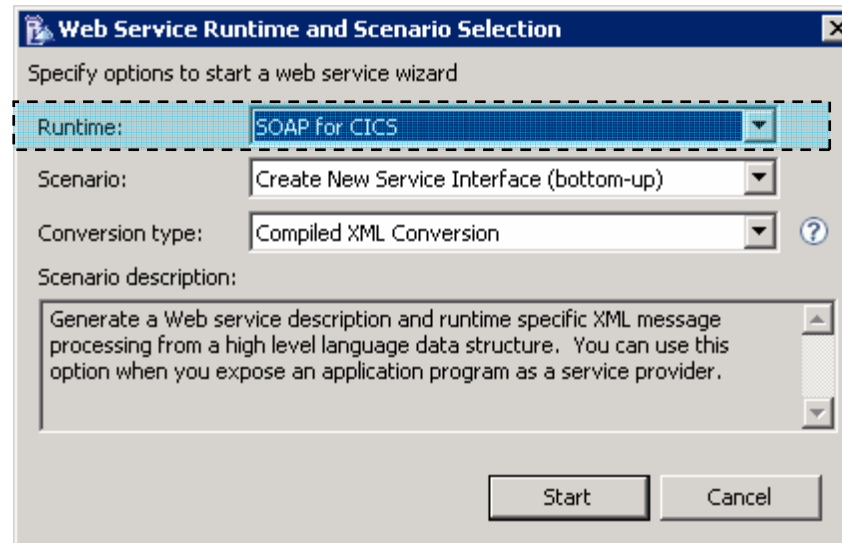
© IBM 2007

Web Service Runtime and Scenario Selection *dialog*



Start the appropriate XSE wizard for selected combination of target runtime, web service development scenario and XML conversion technology type.

Web Service Runtime and Scenario Selection *dialog*



Supported Runtimes

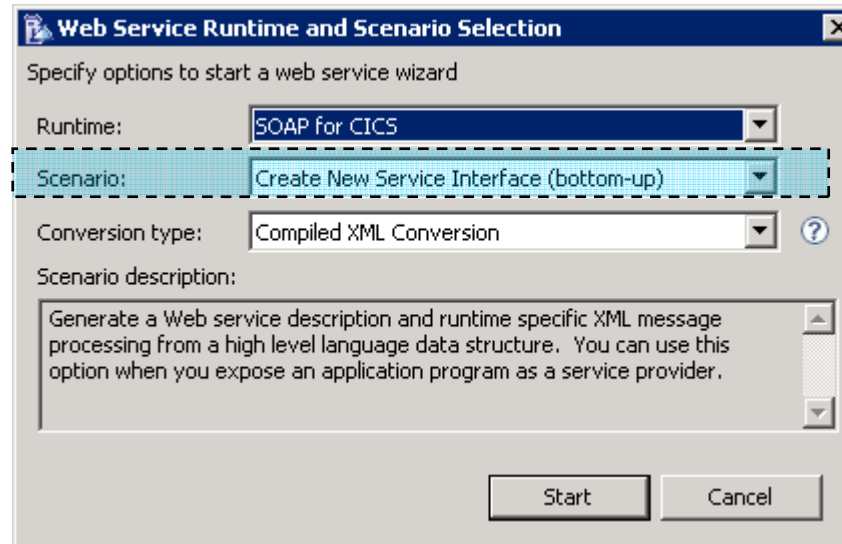
Web Services for CICS (CICS TS 3.1)

SOAP for CICS (CICS TS 2.2/2.3/3.1)

IMS SOAP Gateway

Batch, TSO and USS

Web Service Runtime and Scenario Selection *dialog*



Web Service Runtime and Scenario Selection

Specify options to start a web service wizard

Runtime: SOAP for CICS

Scenario: Create New Service Interface (bottom-up)

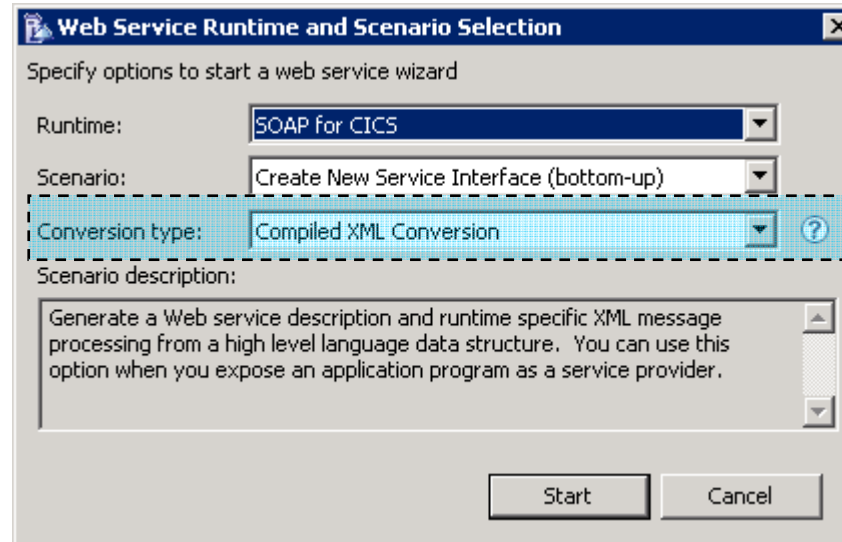
Conversion type: Compiled XML Conversion

Scenario description:
Generate a Web service description and runtime specific XML message processing from a high level language data structure. You can use this option when you expose an application program as a service provider.

Start Cancel

| Scenario name | Scenario nick-name |
|-----------------------------------|--------------------|
| Create New Service Interface | Bottom-up |
| Map an Existing Service Interface | Meet-in-middle |
| Create New Service Implementation | Top-down |

Web Service Runtime and Scenario Selection *dialog*



Supported Conversion Types

Compiled XML Conversion

Interpretive XML Conversion (CICS TS 3.1)



e-business

Web Service Runtime and Scenario Selection *dialog*

Web Service Runtime and Scenario Selection

Specify options to start a web service wizard

Runtime: Web Services for CICS

Scenario: Create New Service Interface (bottom-up)

Conversion type: Compiled XML Conversion

Scenario description: Compiled XML Conversion
Interpretive XML Conversion

Generate a Web service description and runtime specific XML message processing from a high level language data structure. You can use this option when you expose an application program as a service provider.

Start Cancel

Help on the XML Conversion technology type is available. Interpretive is *only* available for Web Services for CICS runtime.

Compiled XML Conversion:

XML conversion is accomplished using a suite of generated high-level language (HLL) programs. A driver program interacts with the runtime and invokes bundled programs to provide conversion of XML to and from language structures. The Compiled XML Conversion type provides more extensive support for language structure data types and constructs than the Interpretive XML Conversion type, however compilation of XML Converters is required and there are more artifacts to manage.

Interpretive XML Conversion:

XML conversion is accomplished using generated metadata and an XML conversion component built into the runtime. While the Interpretive XML Conversion type has limited support for language structure data types and constructs its requires fewer artifacts and does not involve compilation of XML Converters.



[Introduction to XML Services for the Enterprise](#)

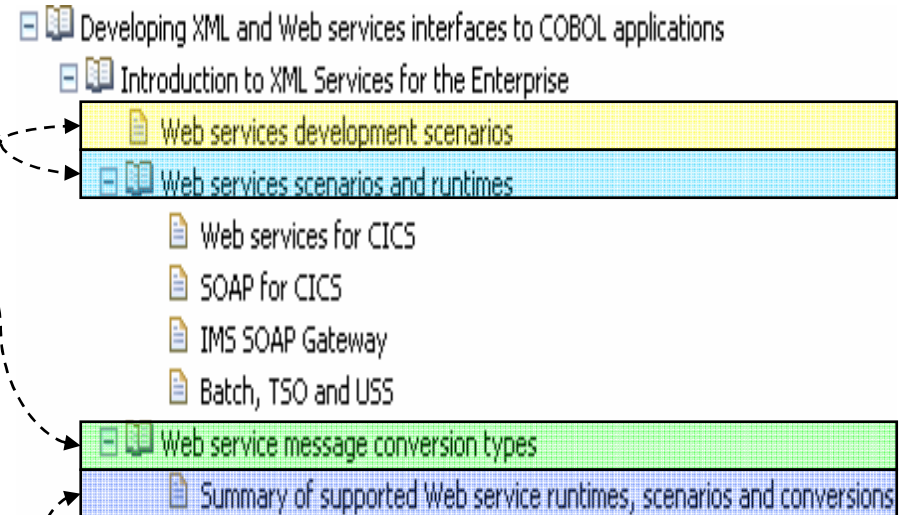
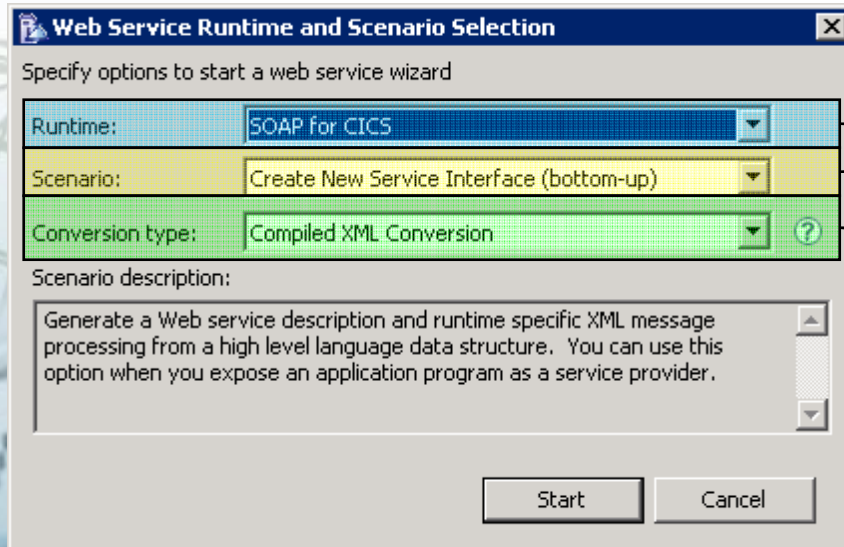
IBM

© IBM 2007



e-business

Web Service Runtime and Scenario Selection *dialog*



To learn about the Runtimes, Scenarios and Conversion types, refer to the WD/z online help. Note that not all possible combinations are supported. The WD/z online help has a

© IBM 2007



e-business

Create New Service Interface (bottom-up) wizard

Compiled XML conversion type

Common pages

- SOAP for CICS
- Batch, TSO, and USS

- Web services for CICS
- IMS SOAP Gateway

Web Services for CICS - Create New Service Interface (betto...)

Language structures

The language structures have been imported from the language source. Select the inbound and outbound language structures.

Inbound language structure Outbound language structure

Select the language structure for the inbound XML converter

- SQLCA
- THP
- SQL-MESSAGE
- SQLERROR
- SQLSTAT
- SQLERRCP
- ABEND-MESSAGE
- HI DATA
- DPDCOMAREA

Change COBOL Options...

< Back Next > Finish Cancel

Web Services for CICS - Create New Service Interface (betto...)

Generation options

Specify generation options for the Web Services enablement artifacts.

XML Converter Options RISDL and XSD Options

Specify identification attributes:

Converter program name prefix: ACTD

Author name: R0D42

Service program name: ACTD

Specify character encodings:

Inbound code page: 1140 USA, Canada, etc. Euro O

Host code page: 1140 USA, Canada, etc. Euro O

Outbound code page: 1140 USA, Canada, etc. Euro O

< Back Next > Finish Cancel

RUNTIME SPECIFIC PAGES

Web Services for CICS - Create New Service Interface (betto...)

File, data set, or member selection

Select the source and targets for the Web Services enablement artifacts.

XML Converters RISDL and XSD

Select targets for the XML Converters and Converter Driver

Converter file container: /Dena

Converter driver file name: ACTD00 .cbl

Inbound Converter file name: ACTD00 .cbl

Outbound Converter file name: ACTD00 .cbl

Generate all to driver

Overwrite files without warning

< Back Next > Finish Cancel





Create New Service Interface (bottom-up) wizard

— Compiled XML conversion type
— **Runtime specific pages**

▪ Web services for CICS

The screenshot shows the 'Web Services for CICS' wizard window. It has a title bar 'Web Services for CICS - Create New Service Interface (bottom-up)'. The main area is titled 'Web Services for CICS' and 'Specify properties of the Web Services Bind file (WSBind)'. There are two tabs: 'WSBind Properties' (selected) and 'Advanced WSBind Properties'. Under 'Specify targets for the WSBind file', there are fields for 'WSBind file container' (value: /Demo), 'WSBind file name' (value: ACTD), and a checkbox for 'Overwrite WSBind file'. Below that is a section for 'Specify CICS application program properties' with a dropdown for 'Program interface' (value: COMMAREA) and a text field for 'Container name'. At the bottom are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

▪ IMS SOAP Gateway

The screenshot shows the 'IMS SOAP Gateway' wizard window. It has a title bar 'IMS SOAP Gateway - Create New Service Interface (bottom-up)'. The main area is titled 'IMS SOAP Gateway' and 'Specify IMS SOAP Gateway Correlator properties'. There is a tab 'Correlator Properties'. Under 'Specify SOAP properties', there is a text field for 'SOAPAction' (value: urn:test1). Below that is a section for 'Specify targets for the correlator' with fields for 'Correlator file container' (value: /test), 'Correlator file name' (value: test1), and a checkbox for 'Overwrite correlator file'. At the bottom is a section for 'Specify interaction properties' with fields for 'Socket timeout' (value: 0), 'Execution timeout' (value: 0), 'LTERM name', 'Connection bundle name' (value: corbundle1), and 'Adapter type' (value: IBM COBOL XML Adapter). At the bottom are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.



Create New Service Interface (bottom-up) wizard

Interpretive XML conversion type

CICS Web Services Assistant pages (bottom up)

Web Services for CICS - Create New Service Interface (bottom-up)

CICS Web Services Assistant
Specify service provider application properties

Specify service interface properties

Inbound language structure: SQLCA

Outbound language structure: SQLCA

Local URI: /ACTD

Specify application properties

Program name: ACTD

Program interface: COMMAREA

Container name:

< Back Next > Finish Cancel

Web Services for CICS - Create New Service Interface (bottom-up)

CICS Web Services Assistant
Specify targets for WSBIND and WSDL files

Specify file container and file names

File container: /test Browse...

WSDL file name: ACTD .wsdl

WSBIND file name: ACTD .wsbind

Overwrite files

< Back Next > Finish Cancel





Create New Service Implementation (top-down) wizard

- Interpretive XML conversion type
- CICS Web Services Assistant pages (top-down)

Web Services for CICS - Create New Service Implementation...
CICS Web Services Assistant
Specify new service provider or requestor properties

Application Properties | Service Properties

Specify new service application properties

Application type: Service Provider
Application language: COBOL
Program name: ACTD
Program interface: CHANNEL
Container name: ACTD

< Back | Next > | Finish | Cancel

Web Services for CICS - Create New Service Implementation...
CICS Web Services Assistant
Specify targets for the new service application artifacts

Language Structures | Web Service Bind | Application Template

Specify targets for the generated language structures

File container: /test
Request file prefix: ACTDE
Response file prefix: ACTDO

Specifies a 1 - 6 character prefix used to generate files that will contain the high level language structures for the application.

Overwrite files

< Back | Next > | Finish | Cancel

Web Services for CICS - Create New Service Implementation...
CICS Web Services Assistant
Specify targets for the new service application artifacts

Language Structures | Web Service Bind | Application Template

Specify targets for new service application template

File container: /test
Template file name: ACTD

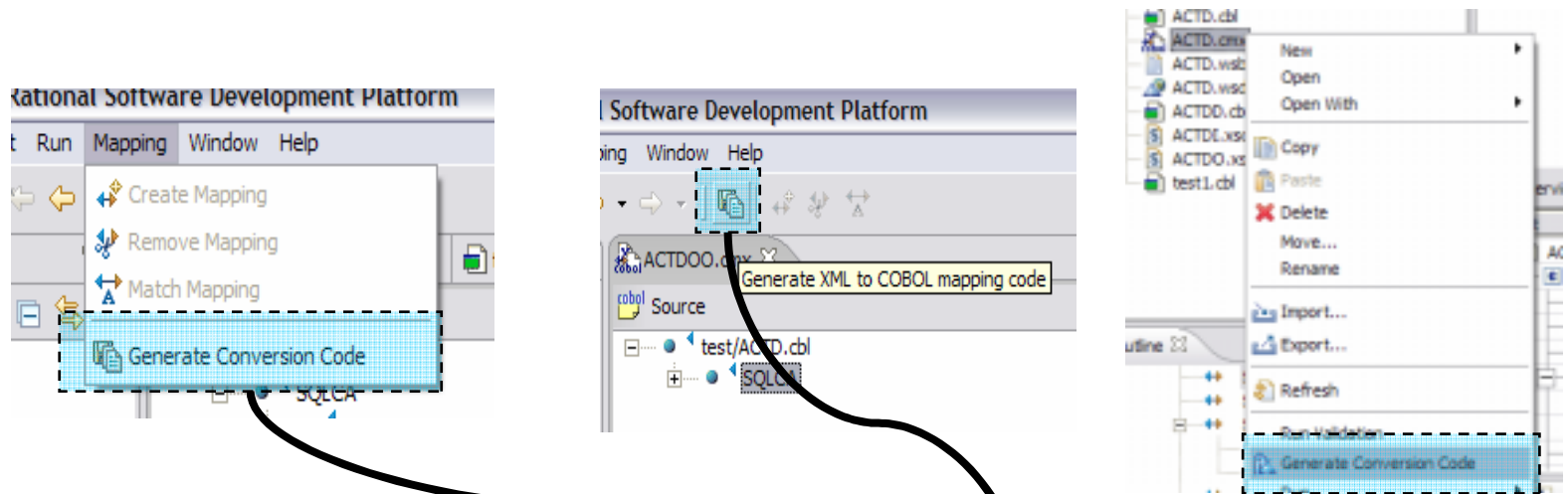
Overwrite files

< Back | Next > | Finish | Cancel



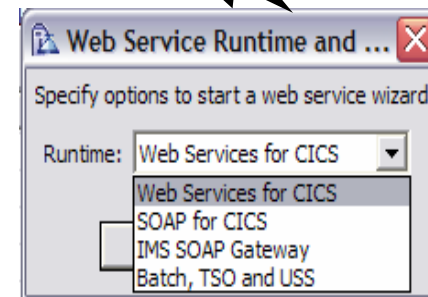
Map to an Existing Service Interface (meet-in-the-middle)

Generate Conversion Code for mapping



You start **Mapping Converter Generator** wizard from **Web Service Runtime selection dialog** (similar to the **Create New Service Interface (bottom-up)** wizard)

Even though you specify the Runtime when you create the Mapping session file, that information does not persist after creation of the mapping file.





Map to an Existing Service Interface (meet-in-the-middle)

Mapping Converter Generator wizard (Outbound mapping)

Web services for CICS

Common

XML to COBOL mapping generation options
Specify XML to COBOL mapping generation options

XML Converter Options

Specify identification attributes

Converter program name prefix: ACTD

Author name: WDHZ

Service program name: ACTD

Specify character encodings

Outbound code page: 1140 USA, Canada, etc. Euro [XXX options...]

Host code page: 1140 USA, Canada, etc. Euro [XXX options...]

Web Services for CICS
Specify properties of the Web Services Bind file (WSBind)

WSBind Properties | Advanced WSBind Properties

Specify targets for the WSBind file

WSBind file container: /test [Browse...]

WSBind file name: ACTD [..WSBind]

Overwrite WSBind file

Specify CICS application program properties

Program interface: COMMERCE

Container name:

- SOAP for CICS
- Batch, TSO, and USS

Common

File, data set, or member selection
Select the source and targets for the Web Services enablement artifacts

XML Converters

Select targets for the XML Converters and Converter Driver

Converter file container: /test [Browse...]

Converter driver file name: ACTD [..dfl]

Outbound Converter file name: ACTD [..dfl]

Generate all to driver

Overwrite files without warning

IMS SOAP Gateway
Specify IMS SOAP Gateway Converter properties

Converter Properties

Specify SOAP properties

SOAPAction: urn:ACTD

Specify targets for the converter

Converter file container: /test [Browse...]

Converter file name: ACTD [..xml]

Overwrite converter file

Specify interaction properties

Socket timeout: 0 (in milliseconds)

Executor timeout: 0 (in milliseconds)

URLName:

Converter bundle name: Combundle1

Adapter type: IBM COBOL XML Adapter

IMS SOAP Gateway





Map to an Existing Service Interface (meet-in-the-middle)

Mapping Converter Generator wizard (Inbound mapping)

Web services for CICS

Common

XML to COBOL mapping generation options
Specify XML to COBOL mapping generation options

XML Converter Options

Specify identification attributes:
Converter program name prefix: ACTDI
Author name: WDMZ
Service program name: ACTDI

Specify character encodings:
Inbound code page: 1140 USA, Canada, etc. Euro
Host code page: 1140 USA, Canada, etc. Euro
Outbound code page: 1140 USA, Canada, etc. Euro

Web Services for CICS
Specify properties of the Web Services Bind file (WSBind)

Specify targets for the WSBind file:
WSBind file container: /test
WSBind file name: ACTDI
WSBind file extension: .wsbind

Specify CICS application program properties:
Program interface: COMMERCE
Container name:

- SOAP for CICS
- Batch, TSO, and USS

Common

File, data set, or member selection
Select the source and targets for the Web Services enablement artifacts.

XML Converters

Select targets for the XML Converters and Converter Driver:
Converter file container: /test
Converter driver file name: ACTDI
Inbound Converter file name: ACTDI
Generate all to driver

IMS SOAP Gateway

IMS SOAP Gateway
Specify IMS SOAP Gateway Converter properties

Specify SOAP properties:
SOAPAction: /urn:ACTDI

Specify targets for the converter:
Converter file container: /test
Converter file name: ACTDI
Converter file extension: .xml

Specify interaction properties:
Socket timeout: (in milliseconds)
Executor timeout: (in milliseconds)
LTPP Name:
Converter bundle name: Commerce 1
Adapter type: IBM CICS XML Adapter





e-business

Example XML and COBOL data structure

```

<?xml version="1.0"?>
<usedcar>
  <price>9,347.99</price>
  <make>AMC</make>
  <model>Gremlin</model>
  <vin>GV39JFGLKM09Y</vin>
  <color>Red</color>
  <mileage>39,000</mileage>
  <inscode>I</inscode>
  <numclaims>1</numclaims>
  <claim>
    <claimno>S8430M4D20030226</claimno>
    <claimamt>1,234</claimamt>
    <insurer>IndemnitiesAreWe</insurer>
    <details>Engine fell out</details>
  </claim>
</usedcar>

```

```

01 USED CAR.
05 PRICE COMP-3 PIC 9(5)V99.
05 SUMMARY.
10 MAKE PIC X(36).
10 MODEL PIC X(44).
10 VIN PIC X(13).
10 COLOR PIC X(10).
88 RED VALUE 'Red'.
88 WHITE VALUE 'White'.
88 BLUE VALUE 'Blue'.
05 HISTORY.
10 MILEAGE PIC 9(6).
10 INSCODE PIC X.
10 NUMCLAIMS BINARY PIC 9.
10 CLAIMS.
15 CLAIM OCCURS 0 TO 9 TIMES
   DEPENDING ON NUMCLAIMS.
20 CLAIMNO PIC X(16).
20 CLAIMAMT BINARY PIC 9(5).
20 INSURER PIC X(39).
20 DETAILS PIC X(100).

```

Used car information:

| Price | Make | Model | VIN | Color | Mileage | InsCode | NumClaims |
|---------|------------------|----------|------------------|-----------------|---------|---------|-----------|
| 0934799 | AMC | Gremlin | GV39JFGLKM09Y | Red | 039000 | I | 1 |
| | ClaimNo | ClaimAmt | Insurer | Details | | | |
| | S8430M4D20030226 | 01234 | IndemnitiesAreWe | Engine fell out | | | |



© IBM 2007



Pros and cons: WD XSE versus direct COBOL coding

■ WD XSE—strengths:

- Interactive tool avoids intricate and laborious programming
 - ▶ Easy to “reface” existing COBOL applications to support XML
- Wholesale conversion between documents and structures
 - ▶ inbound and outbound
- Robust error recovery, using LE services
- Highly optimized and sophisticated content processing
 - ▶ Uses the native z/OS high-speed parser
- Unnecessary elements are conveniently ignored
- Can derive XML definition from COBOL, or match independent XML and COBOL definitions
- Generated schema can be used to validate messages and as input to the WSDL file



Pros and cons: WD XSE versus direct COBOL coding...

■ XML PARSE—strengths:

- Flexibility: all parts of an XML document are accessible:
 - ▶ Attributes, processing instructions, comments, etc.
- Business logic can be conveniently applied during and after message acquisition/generation
- XML definition can be independent of (any) data structure
- Can “short-circuit” parsing early after required input is seen

■ XML GENERATE...FROM—strengths:

- Very simple to use
- A single COBOL statement provides wholesale conversion from a data structure to a document
- The generated XML precisely matches the data structure
- Redefinition allows selective output, different tag names



Agenda...

- Some XML terminology
- Scenarios
- XML support in COBOL
- WebSphere Studio—XML Enablement
- **Discussion, questions**





Sample XML parse program...

```
Process flag(i,i)
Identification division.
    Program-id. xmldump.

Environment division.
    Input-output section.
        File-control.
            select xf assign fi file status fs.

Data division.
    File section.
        FD xf recording mode v
            record varying from 1 to 1024 characters depending on 1.
        1 r.
        2 pic x occurs 1 to 1024 times depending on 1.

Working-storage section.
    1 ft.
        2 pic x(5) value ' ) SHR'.
        2 pic x value low-value.

Local-storage section.
    1 fv.
        2 pic x(7) value 'FI=DSN('.
        2 fn pic x(150).
    1 in-len binary pic 999.
    1 st pic x(9).
    1 rc binary pic 9(9).
    1 l binary pic 9(5).
    1 p binary pic 9(5).
    1 fs pic 99.
    1 xml-document pic x(32767).
```





e-business



© IBM 2007

```
Procedure division.  
mainline section.  
  perform get-doc  
  perform until in-len = 0  
    if p > 0  
      xml parse xml-document(1:p)  
      processing procedure xml-handler  
      on exception  
        display 'XML document error ' xml-code  
      not on exception  
        display 'XML document successfully parsed'  
      end-xml  
    end-if  
  perform get-doc  
end-perform  
  
goback  
.  
  
get-fn section.  
  display  
    ' Enter XML document file name or SYSIN (null to end)'  
  move space to fn  
  move 0 to tally  
  accept fn  
  inspect function reverse(fn) tallying tally  
    for leading space  
  compute in-len = 150 - tally  
  .
```



e-business



© IBM 2007

```
get-doc section.
perform get-fn
evaluate true
  when in-len = 0
    continue
  when fn(1:in-len) = 'SYSIN'
    display ' Enter XML document:'
    move spaces to xml-document(1:150)
    accept xml-document(1:150)
    move function lower-case(xml-document(1:150))
      to xml-document(1:150)
    move 0 to p
    inspect function reverse(xml-document(1:150)) tallying p
      for leading spaces
    compute p = 150 - p
  when other
    move ft to fn(in-len + 1:length of ft)
    call 'putenv' using by value address of fv returning rc
    if rc not = 0
      display 'putenv failed with rc = ' rc '.'
      stop run
    end-if
    open input xf
    if fs = 0
      read xf
    end-if
    move 1 to p
    perform until fs not = 0
      if p - 1 + length of r > length of xml-document
        display 'XML document is larger than the document '
          'buffer (' length of xml-document ' bytes).'
        move 13 to fs
      else
        string r delimited by size into xml-document
          with pointer p
        read xf
      end-if
    end-perform
    evaluate fs
    when 10
      subtract 1 from p
    when 13
      move 0 to p
    when other
      display 'Some catastrophe on file ' fn(1:in-len)
        '; status = ' fs '.'
      move 0 to p
    end-evaluate
    close xf
  end-evaluate
end-evaluate
.
```



e-business



© IBM 2007

```
xml-handler section.  
evaluate xml-event  
  when 'START-OF-DOCUMENT'  
    compute rc = function length(xml-text)  
    move rc to st  
    call 'nzp' using st 1  
    display ' '  
    display 'Start of document: length=' st(1:)  
      ' characters.'  
  when 'END-OF-DOCUMENT'  
    display 'End of document.'  
    display ' '  
  when 'VERSION-INFORMATION'  
    display 'Version: <' xml-text '>'  
  when 'ENCODING-DECLARATION'  
    display 'Encoding: <' xml-text '>'  
  when 'STANDALONE-DECLARATION'  
    display 'Standalone: <' xml-text '>'  
  when 'START-OF-ELEMENT'  
    display 'Start element tag: <' xml-text '>'  
  when 'ATTRIBUTE-NAME'  
    display 'Attribute name: <' xml-text '>'  
  when 'ATTRIBUTE-CHARACTERS'  
    display 'Attribute value characters: <' xml-text '>'  
  when 'ATTRIBUTE-CHARACTER'  
    display 'Attribute value character: <' xml-text '>'  
  when 'END-OF-ELEMENT'  
    display 'End element tag: <' xml-text '>'  
  when 'START-OF-CDATA-SECTION'  
    display 'Start of CData: <' xml-text '>'  
  when 'END-OF-CDATA-SECTION'  
    display 'End of CData: <' xml-text '>'  
  when 'CONTENT-CHARACTERS'  
    display 'Content characters: <' xml-text '>'  
  when 'CONTENT-CHARACTER'  
    display 'Content character: <' xml-text '>'  
  when 'PROCESSING-INSTRUCTION-TARGET'  
    display 'PI target: <' xml-text '>'  
  when 'PROCESSING-INSTRUCTION-DATA'  
    display 'PI data: <' xml-text '>'  
  when 'COMMENT'  
    display 'Comment: <' xml-text '>'  
  when 'EXCEPTION'  
    compute rc = function length (xml-text)  
    move rc to st  
    call 'nzp' using st 1  
    display 'Exception ' xml-code ' at offset ' st(1:) '.'  
  when other  
    display 'Unexpected xml event: ' xml-event '.'  
end-evaluate  
.  
End program xmldump.
```



e-business



Identification division.

Program-id.

nzp.

Data division.

Linkage section.

1 str pic x(9).

1 pos binary pic 9(5).

Procedure division using str pos.

if str = '000000000'

move 9 to pos

else

move 0 to pos

inspect str tallying pos for leading '0'

add 1 to pos

end-if

goback

.

End program nzp.

IBM

© IBM 2007



Sample XML generate program...

Identification division.

Program-id. XGFX.

Data division.

Working-storage section.

01 numItems pic 99 global.

01 purchaseOrder global.

05 orderDate pic x(10).

05 shipTo.

10 country pic xx value 'US'.

10 name pic x(30).

10 street pic x(30).

10 city pic x(30).

10 state pic xx.

10 zip pic x(10).

05 billTo.

10 country pic xx value 'US'.

10 name pic x(30).

10 street pic x(30).

10 city pic x(30).

10 state pic xx.

10 zip pic x(10).

05 orderComment pic x(80).

05 items.

10 item occurs 0 to 20 times depending on numItems.

15 partNum pic x(6).

15 productName pic x(50).

15 quantity pic 99.

15 USPrice pic 999v99.

15 shipDate pic x(10).

15 itemComment pic x(40).

01 numChars comp pic 999.

01 xmlPO pic x(999).





e-business



© IBM 2007

Procedure division.

m.

Move 20 to numItems

Move spaces to purchaseOrder

Move '1999-10-20' to orderDate

Move 'US' to country of shipTo

Move 'Alice Smith' to name of shipTo

Move '123 Maple Street' to street of shipTo

Move 'Mill Valley' to city of shipTo

Move 'CA' to state of shipTo

Move '90952' to zip of shipTo

Move 'US' to country of billTo

Move 'Robert Smith' to name of billTo

Move '8 Oak Avenue' to street of billTo

Move 'Old Town' to city of billTo

Move 'PA' to state of billTo

Move '95819' to zip of billTo

Move 'Hurry, my lawn is going wild!' to orderComment

Move 0 to numItems

Call 'addFirstItem'

Call 'addSecondItem'

Move space to xmlPO

Xml generate xmlPO from purchaseOrder count in numChars

Call 'pretty' using xmlPO value numChars

Goback.



e-business



```
Identification division.  
  Program-id. 'addFirstItem'.  
Procedure division.  
  Add 1 to numItems  
  Move '872-AA' to partNum(numItems)  
  Move 'Lawnmower' to productName(numItems)  
  Move 1 to quantity(numItems)  
  Move 148.95 to USPrice(numItems)  
  Move 'Confirm this is electric' to itemComment(numItems)  
  Goback.  
End program 'addFirstItem'.  
  
Identification division.  
  Program-id. 'addSecondItem'.  
Procedure division.  
  Add 1 to numItems  
  Move '926-AA' to partNum(numItems)  
  Move 'Baby Monitor' to productName(numItems)  
  Move 1 to quantity(numItems)  
  Move 39.98 to USPrice(numItems)  
  Move '1999-05-21' to shipDate(numItems)  
  Goback.  
End program 'addSecondItem'.  
  
End program XGFX.
```



e-business



Identification division.

Program-id. Pretty.

Data division.

Working-storage section.

01 prettyPrint.

05 pose pic 999.

05 posd pic 999.

05 depth pic 99.

05 element pic x(30).

05 indent pic x(20).

05 buffer pic x(100).

Linkage section.

1 doc.

2 pic x occurs 16384 times depending on len.

1 len comp-5 pic 9(9).

Procedure division using doc value len.

m.

Move space to prettyPrint

Move 0 to depth posd

Move 1 to pose

Xml parse doc processing procedure p

Goback.

IBM

© IBM 2007



e-business



IBM

© IBM 2007

```
P.
Evaluate xml-event
  When 'START-OF-ELEMENT'
    If element not = space
      If depth > 1
        Display indent(1:2 * depth - 2) buffer(1:pose - 1)
      Else
        Display buffer(1:pose - 1)
      End-if
    End-if
  Move xml-text to element
  Add 1 to depth
  Move 1 to pose
  String '<' xml-text '>' delimited by size into buffer
    with pointer pose
  Move pose to posd
  When 'CONTENT-CHARACTERS'
    String xml-text delimited by size into buffer
      with pointer posd
  When 'CONTENT-CHARACTER'
    String xml-text delimited by size into buffer
      with pointer posd
  When 'END-OF-ELEMENT'
    Move space to element
    String '</' xml-text '>' delimited by size into buffer
      with pointer posd
    If depth > 1
      Display indent(1:2 * depth - 2) buffer(1:posd - 1)
    Else
      Display buffer(1:posd - 1)
    End-if
    Subtract 1 from depth
    Move 1 to posd
  When other
    Continue
  End-evaluate.
End program Pretty.
```



Output from generate program

```
<purchaseOrder>
  <orderDate>1999-10-20</orderDate>
  <shipTo>
    <country>US</country>
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo>
    <country>US</country>
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <orderComment>Hurry, my lawn is going wild!</orderComment>
```





e-business



```
<items>
  <item>
    <partNum>872-AA</partNum>
    <productName>Lawnmower</productName>
    <quantity>1</quantity>
    <USPrice>148.95</USPrice>
    <shipDate> </shipDate>
    <itemComment>Confirm this is electric</itemComment>
  </item>
  <item>
    <partNum>926-AA</partNum>
    <productName>Baby Monitor</productName>
    <quantity>1</quantity>
    <USPrice>39.98</USPrice>
    <shipDate>1999-05-21</shipDate>
    <itemComment> </itemComment>
  </item>
</items>
</purchaseOrder>
```