XML and Web Services for the Enterprise

Providing XML and Web Services Interface To a CICS Application

Teodoro Cipresso
Anthony Flusche
Gary Mazo

IBM Silicon Valley Laboratory

August 2004

Table of contents

1 Acknowledgements 4
2 Introduction 5
3 Business Application Sample 6
3.1 Parts of the existing business application 6
3.2 Setting up and running the existing business application 7
3.2.1 Preparing your datasets 7
3.2.2 Configuring DB2 7
3.2.3 Assembling BMS maps 7
3.2.4 Pre-compiling the existing programs 9
3.2.5 Compiling and link-editing the sample 9
3.2.6 Binding the DB2 tables 9
3.2.7 Configuring CICS 10
3.2.8 Running the application 10
4 XML-enabled Business Application 12
4.1 Using the Enterprise Developer XML Enablement Tool 13
4.1.1 Locating your existing application 14
4.1.2 Creating a project in Enterprise Developer 15
4.1.3 Moving your existing application to Enterprise Developer 16
4.1.4 Invoking the XML Converter generator wizard 17
4.1.5 Specifying input and output files for the wizard 18
4.1.6 Specifying the generation options 19
4.1.7 Specifying input and output data structures 21
4.1.8 Generating code 23
4.1.9 Generating additional converters 24
4.1.10 Modifying the CICS converter driver program 25
4.2 Running your XML Enabled application on the mainframe 28
4.2.1 Preparing your datasets 28
4.2.2 Configuring DB2 29
4.2.3 Assembling BMS maps 30
4.2.4 Pre-compiling the existing programs 30
4.2.5 Compiling and link-editing the existing application 30
4.2.6 Compiling and link-editing the XML processing code 30
4.2.7 Binding the DB2 tables 31
4.2.8 Configuring CICS 31
4.2.9 Running the application 31
4.2.10 Error reporting 32
5 Enabling your business application for Web Services 33
5.1 Preparing your application to run as a Web Service on the mainframe 33
5.1.1 Locating your existing application 33
5.1.2 Creating an Enterprise Developer project 35
5.1.3 Moving your existing application to Enterprise Developer 36
5.1.4 Invoking the XML Converter generator wizard 37
5.1.5 Specifying input and output files for the wizard 38
5.1.6 Specifying the generation options 39
5.1.7 Specifying input and output data structures 41
5.1.8 Generating code 43
5.1.9 Generating additional converters 44
5.1.10 Modifying the CICS SOAP converter driver programs 44
5.1.11 Preparing your datasets 48
5.1.12 Configuring DB2 49
5.1.13 Pre-compiling the existing programs 50
5.1.14 Compiling and link-editing the existing application 50
5.1.15 Compiling and link-editing the XML processing code 50
5.1.16 Binding the DB2 tables 51
5.1.17 Configuring CICS 51

5.2 Creating a SOAP-Based Web Service client for an Enterprise Application 52
5.3 Using the Batch Processor (XSEBATCH) 64
5.4 Testing your Web Services-enabled business application 65
5.5 SOAP Error reporting 67
6 Appendices 69
6.1 Appendix A. Pre-requisite software 69
6.2 Appendix B. Modifying COBOL Generator preferences 70
6.3 Appendix C. Modifying COBOL Importer preferences 72
6.4 Appendix D. XML Converter Interface 74
6.5 Appendix E. Business applications program source 75
6.5.1 DFH0ACTD 75
6.5.2 DFH0CSTD 77
7 Notices 79
7.1 Programming interface information 80
7.2 Trademarks and service marks 81

1 Acknowledgements

Many thanks to Andy Krasun, Michael D. Connor, Stephen Hancock, John Lawrence, Grant Ward Able, Larry England and Mark Cocker for reviewing the various versions of this paper and providing invaluable comments.

2 Introduction

With the advent of Web Services, users and owners of many mainframe applications that used to rely on just binary interfaces for communication have been trying to harness XML as a new means of information exchange. This approach presents unique opportunities and challenges to the programmers who are trying to efficiently adapt business applications in order to process and produce XML documents with minimal disruptions to the existing system infrastructure. An example of one such infrastructure would be a bank Call Support Center in which operators utilize 3270 terminals to access a mainframe CICS® application that retrieves and updates customer and account information. Creating new points of access to such a system may include the addition of a Web-based interface, an automated Voice Response Unit (VRU), or a Web service capability.

In this paper, we will describe how IBM® WebSphere® Studio Enterprise Developer™ tools help you modernize your Enterprise assets, adapt them to process and produce XML messages, and expose them as viable Web Services.

3 Business Application Sample

Throughout this paper, we will use an existing CICS application that is included with Enterprise Developer. First, you will install and run this application, then you will learn a how to enable this application to process and produce XML messages using Enterprise Developer.

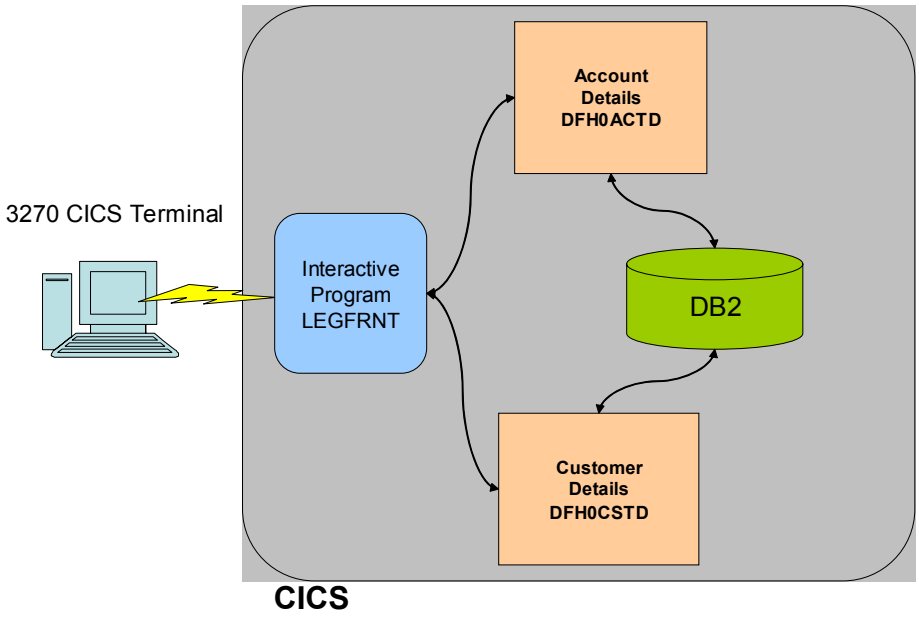


Figure 1, Existing CICS Application

This CICS application consists of an interactive program, LEGFRNT, which calls two CICS programs, DFH0ACTD and DFH0CSTD. They, in turn, access the DB2® table to retrieve customer and account information. This information is exchanged in a binary form via a CICS COMMAREA and the results are displayed on a 3270 terminal. Screens that illustrate the interaction are shown later in this paper.

The rest of this paper assumes that you have some basic familiarity with operations of a CICS environment in OS/390® or z/OS®.

3.1 Parts of the existing business application

Here are the required programs for the current application:

- DFH0ACTD (Account Details sample program)
- DFH0CSTD (Customer Details sample program)
- LEGFRNT (CICS front end program for executing the business programs)
- LEGMAP (CICS BMS map for front end program)
- DFH$EDB2 (creates DB2 tables for the sample programs)
- DFH$ESQL (DB2 bind for the sample programs DFH0ACTD and DFH0CSTD)

- XML$CEDA (creates CICS table entries).

These sample COBOL programs are included with Enterprise Developer. You can find the complete list of pre-requisite software in Appendix A.

3.2 Setting up and running the existing business application

Follow the steps below to set up and run the application.

3.2.1 Preparing your datasets

To set up your application, you allocate the following partitioned data sets (PDS or PDSE) then transfer the associated members. The source files for these programs are shipped in the "Web Services for Enterprise White paper" directory on the Enterprise Developer Installation CD. You can use the IDE for z/OS tools to transfer the source files as members to the appropriate datasets on z/OS.

Data set name	Data set members	Data set characteristics
XML.COBOL	COBOL source code: • DFH0ACTD • DFH0CSTD • LEGFRNT	RECFM: FB LRECL: 80 BLKSIZE: any (e.g. 4000)
XML.CNTL	JCL: • DFH$EDB2 • DFH$ESQL • XML$CEDA	RECFM: FB LRECL: 80 BLKSIZE: any (e.g. 4000)
XML.LOAD	Load modules	RECFM: U LRECL: 0 BLKSIZE: 32760
XML.OBJECT	Object decks	RECFM: FB LRECL: 80 BLKSIZE: any (e.g. 4000)
XML.DBRMLIB	DB2 sample data	RECFM: FB LRECL: 80 BLKSIZE: any (e.g. 4000)
XML.BMS	LEGMAP (BMS map)	RECFM: FB LRECL: 80 BLKSIZE: any (e.g. 4000)

3.2.2 Configuring DB2

Use the DB2 sample program DSNTIAD (which is shipped with DB2) to create the DB2 tables "ACCOUNT" and "CUSTOMER" which are needed by the sample programs. Use DFH$EDB2 as a template for creating the tables. Create the tables by running the DB2 sample program DSNTIAD. The DSN SYSTEM (…) parameter is the name of your DB2 subsystem. This DB2 subsystem should be connected to your target CICS. You should replace DSTNIAxx with the correct name that corresponds to the release level of your DB2 subsystem, for example, DSTNIA61 for DB2 6.1. Note that starting with release 6 of DB2, DSNTIAD is shipped as source and a load module; in prior releases, it is available as source only. The sample JCL to assemble the DSNTIAD source can be found in DB2.SDSNSAMP(DSNTIJTM) as one of the steps in the control file:

```
//SYSTSIN DD *
      DSN SYSTEM(…)
      RUN PROGRAM(DSNTIAD) PLAN(DSNTIAxx)
      END
//SYSIN  DD    DSN=DFH$EDB2,DISP=SHR
```

3.2.3 Assembling BMS maps

To create the layout of the CICS front end to the sample programs, you assemble the BMS map LEGMAP using the procedure DFHMAPT supplied with CICS. The resultant COBOL copybook is referenced in the LEGFRNT program.

3.2.4 Pre-compiling the existing programs

Since the sample programs contain EXEC SQL statements they must be pre-compiled. Use the DB2 pre-compiler DSNHPC to pre-compile the sample programs DFH0ACTD and DFH0CSTD.

3.2.5 Compiling and link-editing the sample

Compile and Link-Edit the sample and front end programs DFH0ACTD, DFH0CSTD and LEGFRNT using the procedure IGYWCL as three separate CICS programs. Be sure to include the CICS compile option. Ensure that the resulting load modules are in a load data set visible to the CICS RPL.

3.2.6 Binding the DB2 tables

To allow the sample programs to access the DB2 tables use the sample DFH$ESQL to perform a DB2 bind for the programs DFH0ACTD and DFH0CSTD. The contents of DFH$ESQL should be as follows:

```
DSN SYSTEM(…)
  BIND PACKAGE(EBUSCOL) -
       OWNER(TONYF) -
       QUALIFIER(TONYF) -
       MEMBER(DFH0CSTD) -
       LIBRARY('TONYF.DBRMLIB.DATA') -
       ACTION(REP) -
       ISOLATION(CS)   -
       VALIDATE(BIND)  -
       DYNAMICRULES(BIND)  -
       ENABLE(CICS)

  BIND PACKAGE(EBUSCOL) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      MEMBER(DFH0ACTD) -
      LIBRARY('TONYF.DBRMLIB.DATA') -
      ACTION(REP) -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)

 BIND PLAN(EBUSPLAN) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      ACTION(REP) -
      PKLIST(EBUSCOL.*)  -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)
   END
 COMMIT;
```

Note that you should use your DB2 subsystem ID as the parameter to the DSN SYSTEM directive. Also replace the highlighted fields with your system's high-level qualifier.

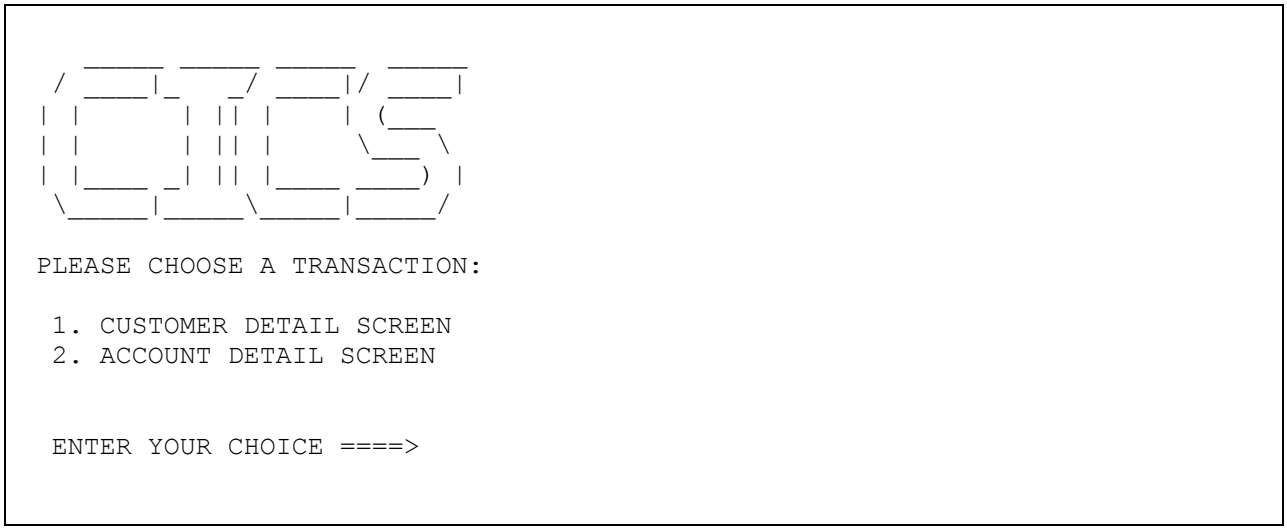
3.2.7 Configuring CICS

Define various resources to CICS. A sample XML$CEDA is provided to assist with this:

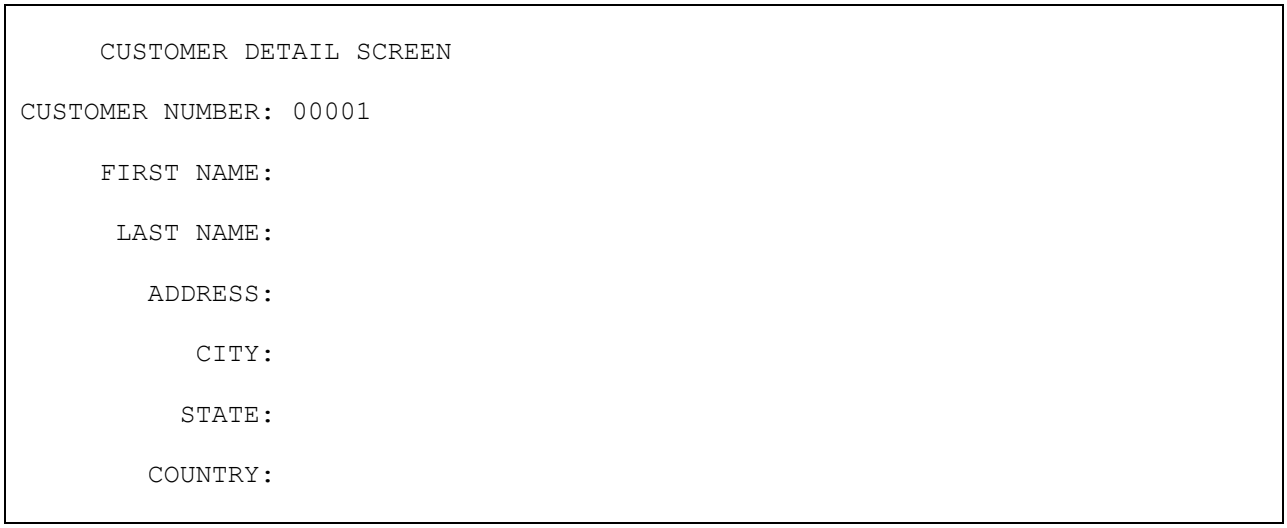
- A transaction named LEGF
- The programs DFH0ACTD, DFH0CSTD and LEGFRNT
- The BMS map LEGMAP
- A DB2ENTRY for LEGF that connects it to sample plan EBUSPLAN
- A DB2 transaction for LEGF.

3.2.8 Running the application

To start the application bring up a CICS terminal and run transaction LEGF. The following screen should appear. Note: the following instructions illustrate transaction 1. The procedure for transaction 2 is the same.



If you enter 1 as your choice the following will appear on the screen:



Change 00001 to 00004 and press Enter. The following screen will appear:

```
    CUSTOMER DETAIL SCREEN

CUSTOMER NUMBER: 00004

     FIRST NAME: JOHN

      LAST NAME: ROYSON

        ADDRESS: 15 MISTYVIEW

           CITY: ROANOKE

          STATE: TX

        COUNTRY: US
```

Note that this program is not a complete CICS application and in order to return to the initial screen you will need to restart the LEGF transaction.

4 XML-enabled Business Application

In order to allow XML documents to flow through to the existing business programs, the source of those programs is passed through the XML Enablement tool in Enterprise Developer. The tool generates a set of COBOL programs called “XML converters” (Inbound and Outbound) based on the original binary interface. The tool also generates a template COBOL program called “Converter driver” that illustrates how to invoke the converters. In section 4 we will show how to augment the driver with EXEC CICS statements to call the existing business application in concert with calling the XML converters. An interactive menu-driven 3270 front-end program facilitates local testing of the new application. The diagram below shows the structure of the modernized application.

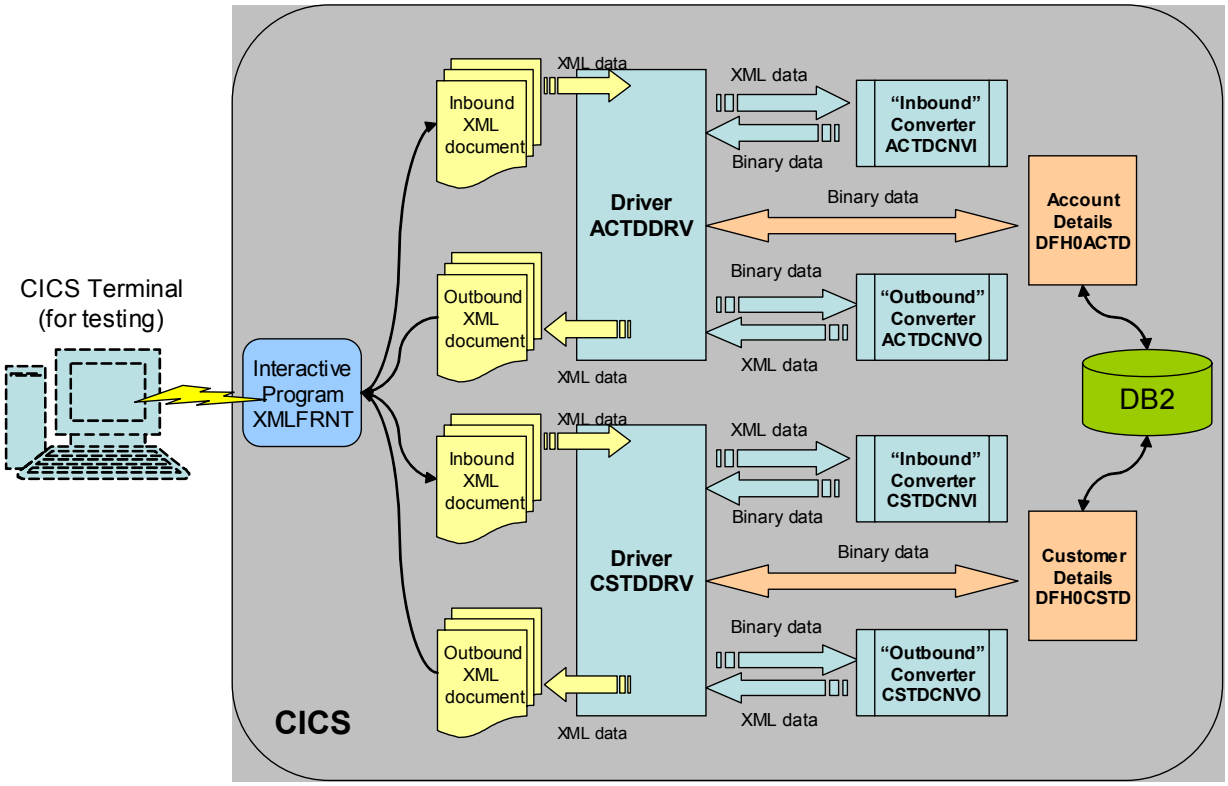


Figure 2, Existing Business CICS Application Enabled for XML

Parts of the application are described in detail later in the paper.

The rest of the paper describes the process that you follow to replace the binary interface of two sample COBOL CICS programs with an XML interface and then how to turn them into full-fledged Web Services.

4.1 Using the Enterprise Developer XML Enablement Tool

This section describes how to use the XML Enablement tool in Enterprise Developer to generate XML Converters and a Driver template. Before you read section 4 you should familiarize yourself with the basic concepts of Enterprise Developer, including its IDE for z/OS feature.

Required programs for this sample:

- **DFH0ACTD** (Account Details sample program)
- **DFH0CSTD** (Customer Details sample program).

These COBOL programs are shipped with Enterprise Developer samples.

During development of your application, you will also use the following programs and datasets:

- **ACTDCNV(I,O)** - account details inbound/outbound XML converters that are generated by the XML Converter generator. The **inbound** XML converter is a COBOL program that processes an incoming XML document and converts contents of its elements into a COBOL data structure. The inbound converter uses high-performance XML parsing capabilities of the latest IBM Enterprise COBOL compiler and runtime to efficiently parse the inbound XML document. During the parse, the inbound XML converter converts XML data and stores it in the existing application's COBOL data structure (in case of a CICS application this data structure is a CICS COMMAREA). The conversion and moving of data is based on proprietary algorithms that provide high efficiency in transforming character data from the XML document into appropriate COBOL data. The **outbound** XML converter is a COBOL program that takes the results of the execution of the existing transactional program and converts COBOL data into an XML message. That message will be returned to the client. In case of an error during execution of the transaction, an XML-based error message will be returned
- **CSTDCNV(I,O)** - customer details inbound/outbound XML converters that are generated by the XML Converter generator
- **ACTDDRV (or ACTDSOAP if you want to enable Web Services)** - account details XML converter driver that is generated by the XML Converter generator. The XML converter **driver** is a COBOL program that shows the invocation sequence for the inbound converter, the existing program and the outbound converter
- **CSTDDRV (or CSTDSOAP if you want to enable Web Services)** - customer details XML converter driver that is generated by the XML Converter generator
- **XMLFRNT** - CICS front-end program for executing sample programs (you don't need this program if you will be using Web Services to access your application)
- **XMLMAP** - CICS BMS map for front-end program (you don't need this program if you will be using Web Services to access your application)
- **DFH$EDB2** - creates DB2 tables for the sample programs
- **DFH$ESQL** - DB2 bind for the sample programs DFH0ACTD and DFH0CSTD
- **XML$CEDA** - creates CICS table entries.

Follow the steps outlined below to create, set up and run your XML-enabled business application.

4.1.1 Locating your existing application

Locate the sources for DFH0ACTD and DFH0CSTD on your z/OS system. In order to do that, you can use IDE for z/OS tools in Enterprise Developer. You define and connect to the remote z/OS system. For this example, let's assume that the system you connect to is called STLABE1. Once you are connected to STLABE1, locate the sources for DFH0ACTD and DFH0CSTD. Let's assume that they are located in a PDS called XML.COBOL under the high level qualifier (HLQ) TONYF.

4.1.2 Creating a project in Enterprise Developer

Create a local container for the generated XML Converters and the Driver template. This local container is called “Simple Project”. In Enterprise Developer, switch to the Resource perspective and invoke the New Project wizard. Select “Simple” as the type of the project you want to create. Follow the wizard to create a project called “XML Account Test” (Fig. 3.)

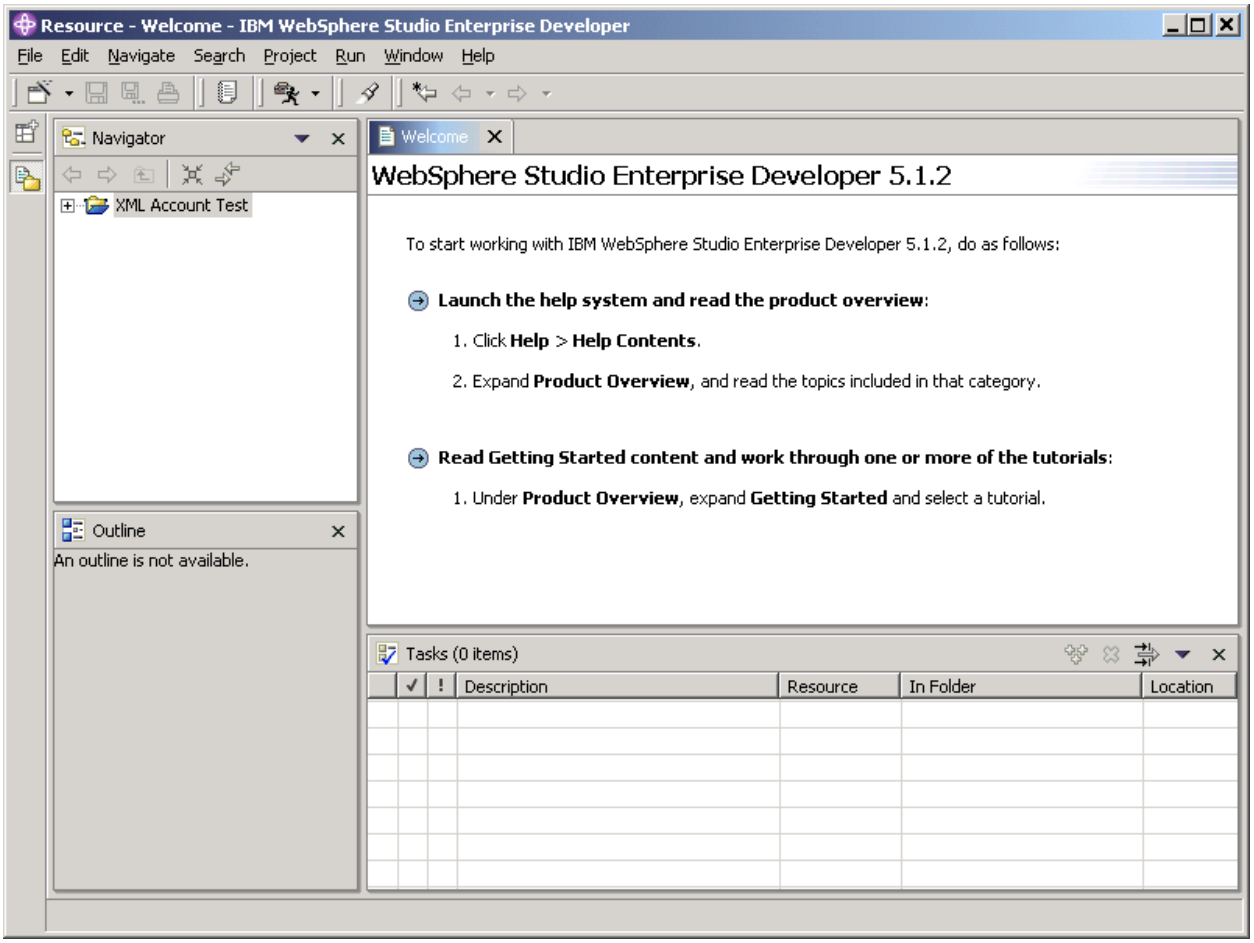


Figure 3, XML Account Test Project

4.1.3 Moving your existing application to Enterprise Developer

Use Copy and Paste operations on the source COBOL files for DFH0ACTD and DFH0CSTD to copy them into your local XML Account Test Project. (Fig. 4) Hint: You use the "Go Into" and "Refresh" actions to "go into" the XML.COBOL PDS represented as a Folder in the Resource perspective.

If the source files are located in the local file system on your PC, use the File> Import operation provided in the Workbench to import the source files into your local project.

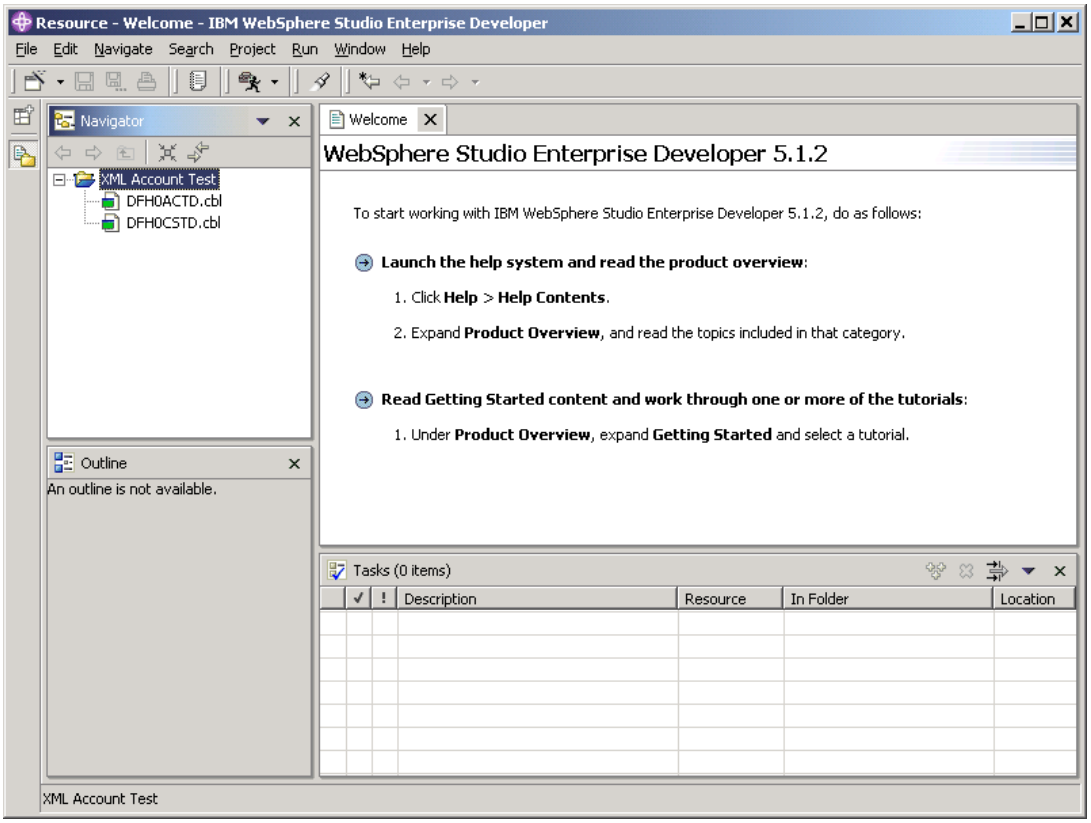


Figure 4, XML Account Test Project With Source

Note that the IDE appends extension .cbl to the source file name. This way you can use file type-specific Enterprise Developer tools like a language-sensitive editor.

4.1.4 Invoking the XML Converter generator wizard

Invoke XML Converter generator for the source program DFH0ACTD.cbl. To do that, select DFH0ACTD.cbl in the Navigator view and invoke the pop-up menu by pressing the right mouse button. (Fig. 5)

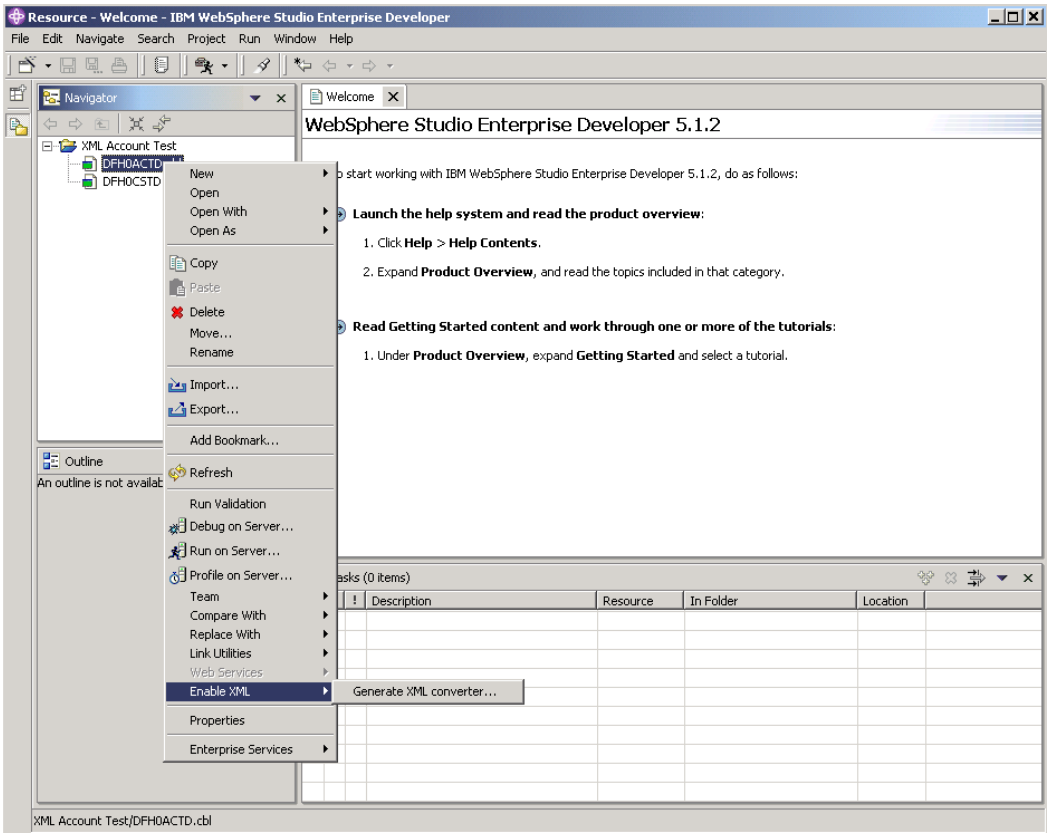
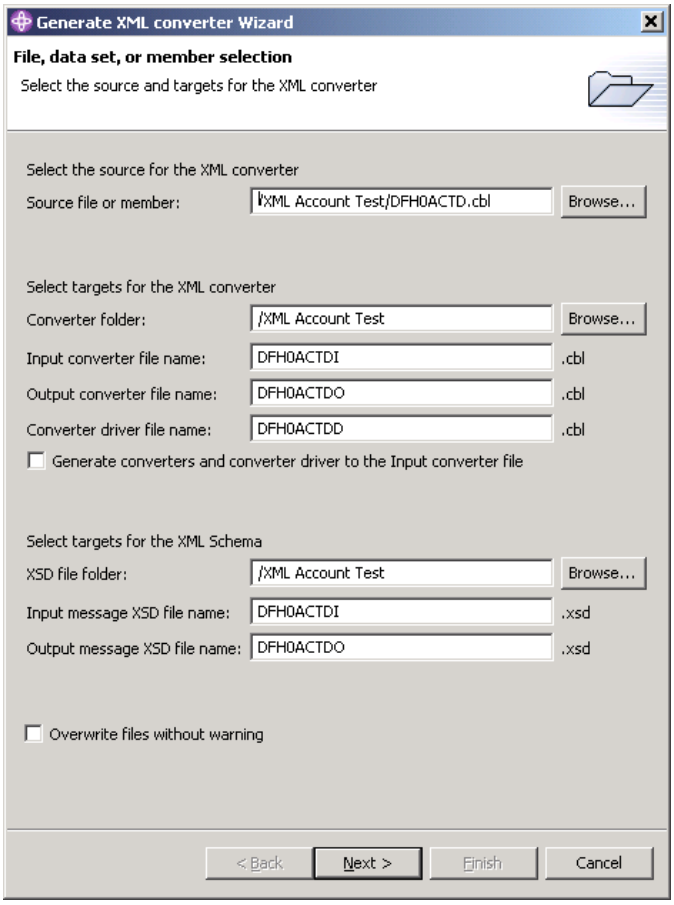


Figure 5, Invoking the XML Converter generator

4.1.5 Specifying input and output files for the wizard

Use the first page of the XML Converter wizard to select input and output files for the Converter, Driver template, and the XML Schema. Hint: The XML Schema is automatically generated that contains the description of the names and types of XML elements. These elements can appear in an XML document that our program will process and generate. For more information on XML Schema visit http://www.w3.org/XML/Schema



CICS XML File Selections

- Input converter file name
  - ACTDCNVI.cbl
- Output converter file name
  - ACTDCNVO.cbl
- Converter driver file name
  - ACTDDRV.cbl

Enter these names in the dialog page shown on the left (Replacing DFH0ACTDI with ACTDCNVI, DFH0ACTDO with ACTDCNVO, and DFH0ACTDD with ACTDDRV)

Figure 6, Generate XML Converter Wizard

On this page, the fields are as follows:

- Source file - specify where your existing COBOL program is located
- Converter folders - specify folder(s) where the wizard will generate converter program(s)
- Converter file names - specify the name(s) you want to give your converter file(s),
- Converter driver file name - specify the name you want to give your converter driver.
- Generate converters and converter driver to one file - select if you want to generate the converters and converter driver to the file specified in the "input converter file name" field. The driver will appear first in the file.
- XSD file folders - specify where the wizard will generate the XML Schema file(s)
- XSD file names - specify the name(s) you want to give your XML Schema file(s)
- Overwrite files without warning - select if you want to overwrite existing output files

4.1.6 Specifying the generation options

Use the second page of the XML Converter wizard to specify generation options for your Converter and Driver programs. Choose the host code page for the Host system were you will deploy the converters. Also set Driver type to CICS. There is no need to modify the rest of the specified defaults on this page. (Fig. 7)

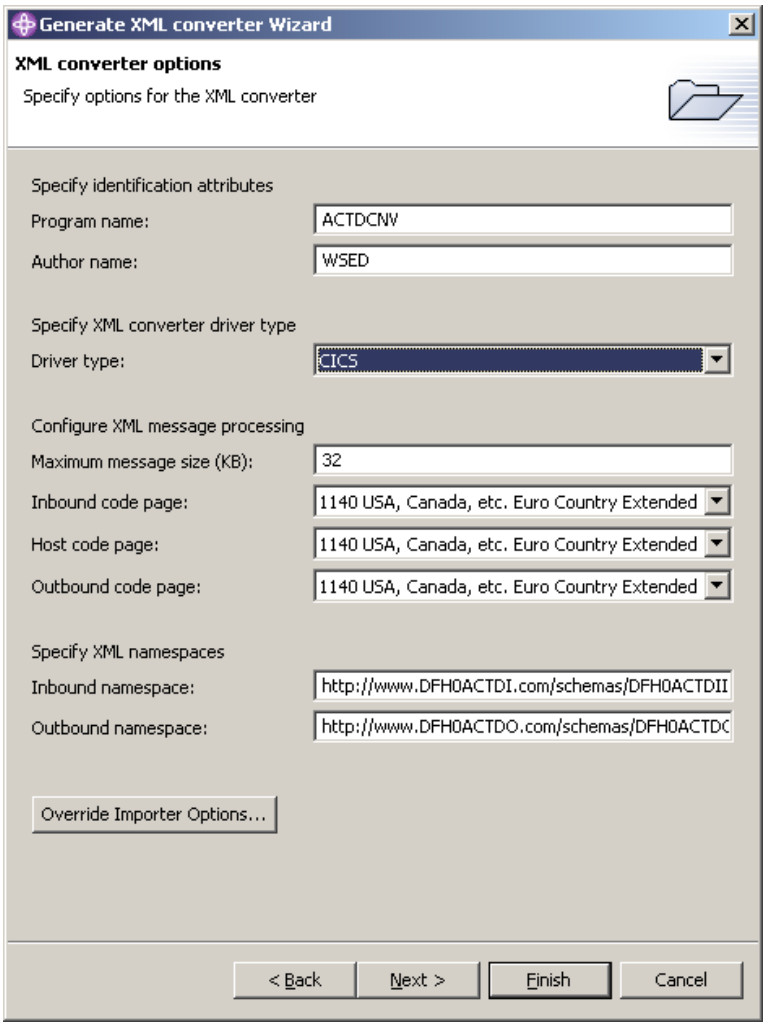


Figure 7, Generation Options Selection

On this page, the fields are as follows:

- Program name – specify the "stem" value for the program names in the PROGRAM-ID paragraph of the IDENTIFICATION DIVISION in the COBOL programs that this wizard will generate. For example, if you enter ACTDCNV, the wizard will generate ACTDCNVI for the inbound converter program name, ACTDCNVO for the outbound converter, and ACTDCNVD for the converter driver.
- Author name – specify the value for the AUTHOR paragraph
- Driver type – specify the desired driver type. Choose "CICS". The CICS driver type provides some conveniences to minimize the need for modifications.
- Maximum message size – specify the maximum size of the XML message that will need to be allocated when processing and generating the XML message.
- Code pages - specify code page(s) for the encoding of the inbound and outbound XML documents, and the code page for the host data.
- Inbound Namespace – specify the inbound namespace or accept the default. This is currently not validated.
- Outbound Namespace – specify the namespace container for messages created by the outbound converter.

Note: The default values for this page are taken from the default preference values stored in the Workbench. You can modify those defaults by visiting the Preference page for the wizard. The process for modifying the preferences is described in “Appendix B. Modifying COBOL Generator preferences” on page 70.

4.1.7 Specifying input and output data structures

Use the third page of the XML Converter wizard to specify the input and output data structures for which you want to generate the equivalent XML-based interface. In the Input and Output data structure tree views, select DFHCOMMAREA as input and output structure. (Fig. 8)

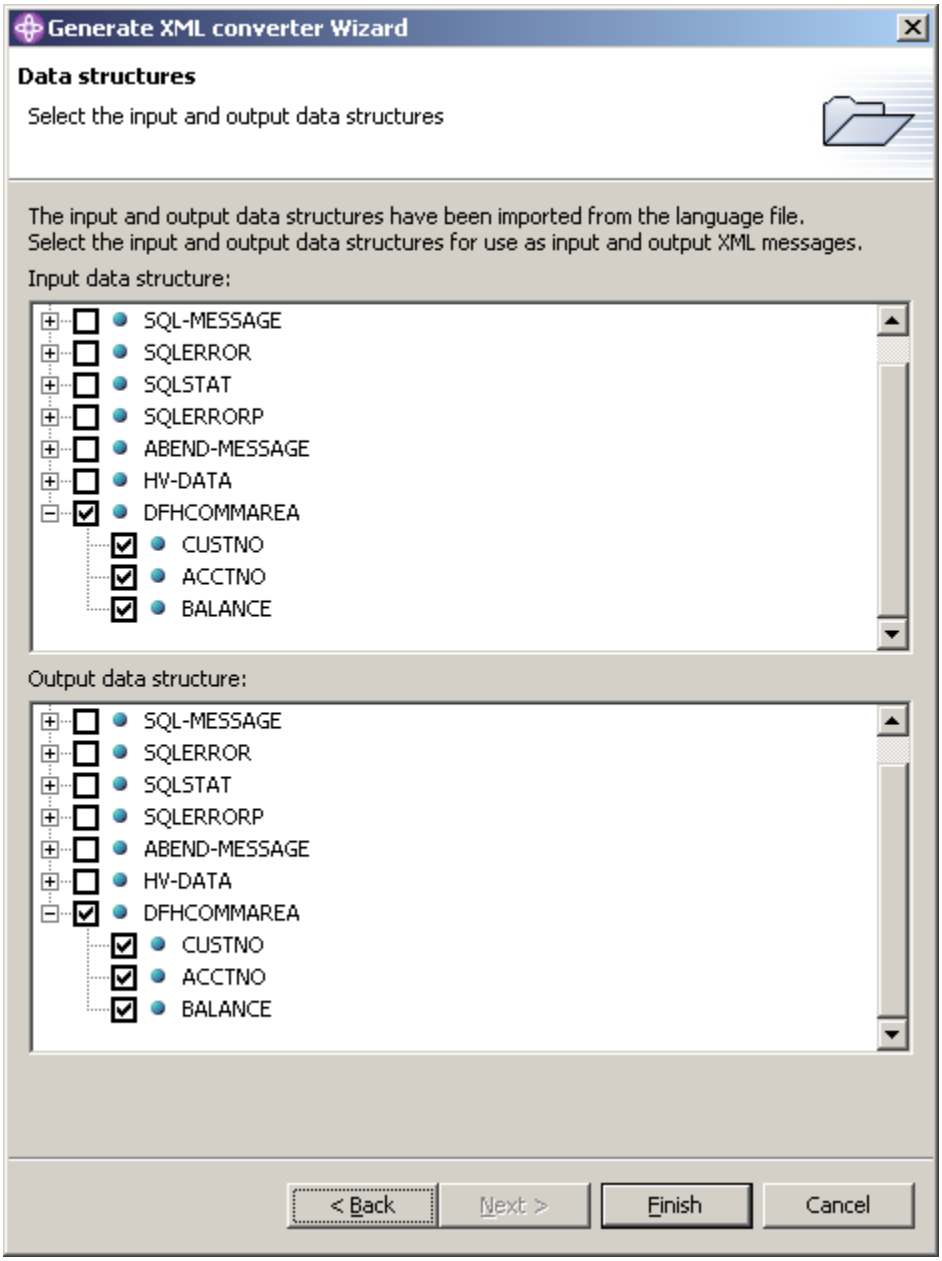


Figure 8, Data Structure Selection

On this page, the fields are as follows:

- `Input data structure` – select the level 01 data from an available list of level 01 COBOL data structures extracted from your existing application program. This data structure will be used to generate inbound XML converters.
- `Output data structure` – same as above but used to generate outbound XML converters.

Note: Some of the attributes for COBOL data items and their treatment by the generated converters depend on COBOL compiler options. An example of such options is the TRUNC(BIN) option that causes the z/OS COBOL compiler to treat all binary items as if they all were native binary items. You can

modify these options using the COBOL Importer preference page. To get to this page, click the “override importer options” button on the generation options page in the wizard. The options set during the wizard session do not persist beyond the current session. To permanently modify the importer preferences refer to Appendix C. Modifying COBOL Importer preferences on page 72.

4.1.8 Generating code

Press Finish to complete the generation process. After the wizard processing completes, you will notice that your Converter and Driver files are generated and displayed in the Navigator view. (Fig. 9)

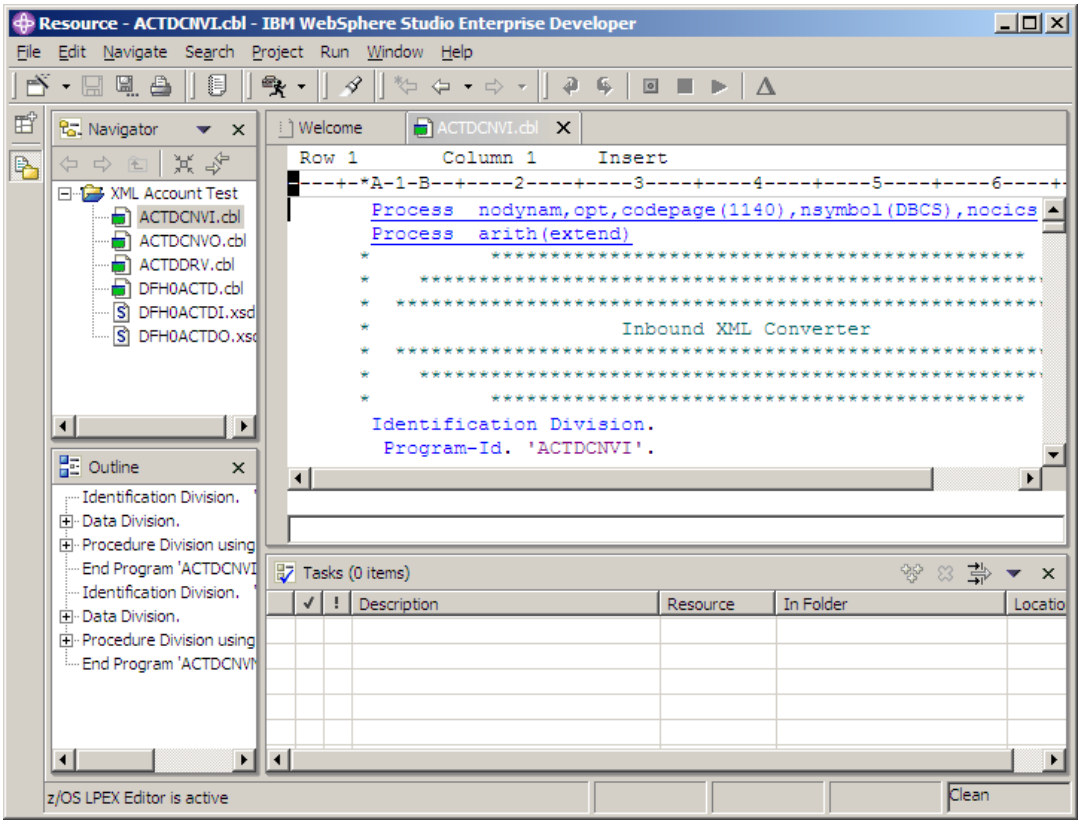


Figure 9, Generated Converters and Driver template for DFH0ACTD

The inbound converter uses the new Enterprise COBOL language, and specifically, the new XML PARSE verb to parse XML documents and convert XML into COBOL data:

```
...
xml parse a-input-xml (1:a-input-xml-len)
       processing procedure a-xml-handler
       thru a-general-logic-exit
      on exception
        perform a-unregister-exception-handler
        perform a-signal-condition
      not on exception
        perform a-unregister-exception-handler
        move zero to a-converter-return-code
  end-xml
...
```

The outbound converter converts the output COBOL data from the existing program into an output XML message:

```
          move CUSTNO
            of DFHCOMMAREA
           to c-CUSTNO-0
            of c-xml-response
```

```
       5 pic x(8) value
            '<custno>'.
       5 c-CUSTNO-0 pic -9(5)
           value zeros.
       5 pic x(9) value
            '</custno>'.
```

4.1.9 Generating additional converters

Use the XML Converter wizard as described previously to generate Converters and Driver template for DFH0CSTD. Notice that Converter and Driver files were generated and displayed in the Navigator view. (Fig. 10)

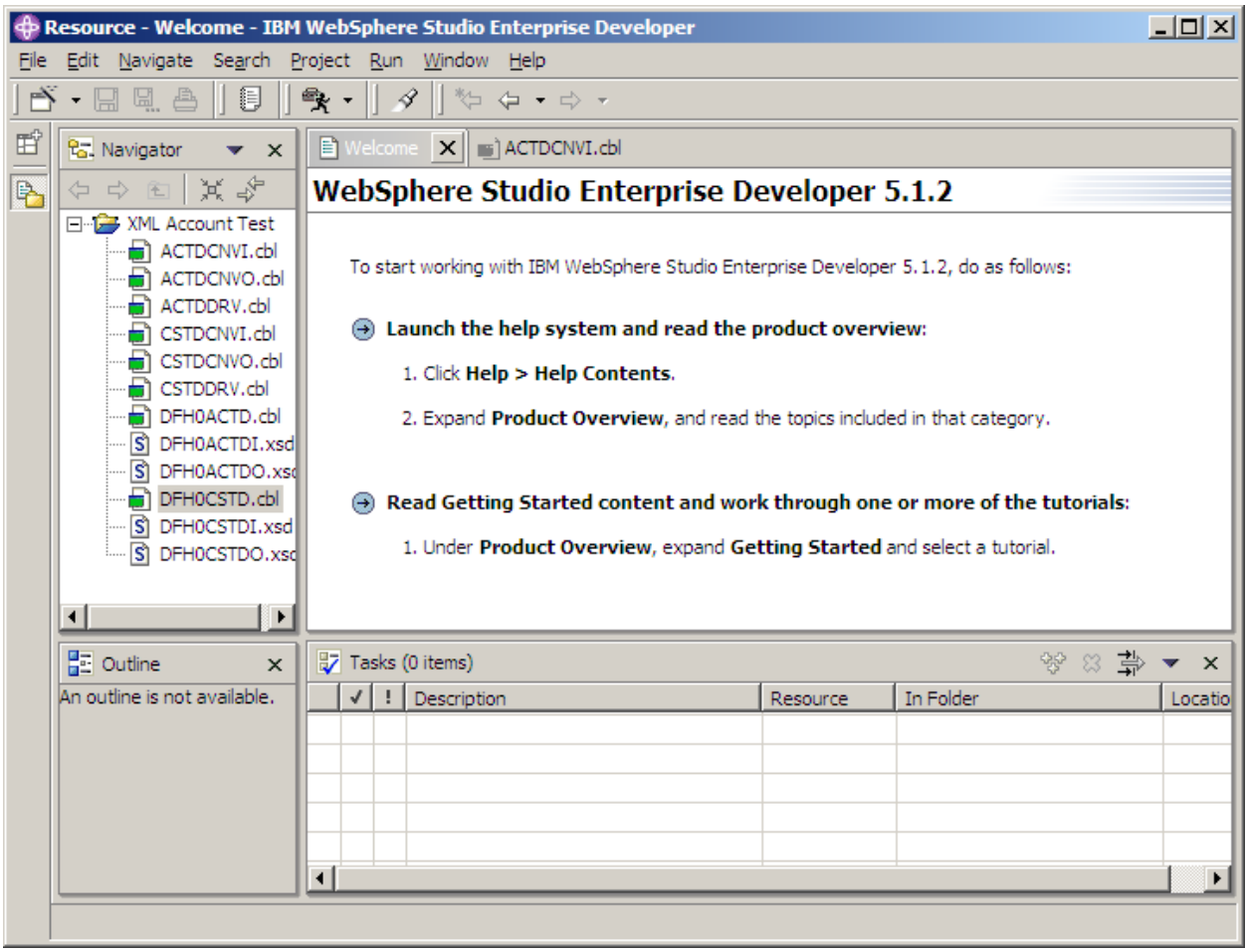


Figure 10, Generated Converters and Driver template for DFH0CSTD

Note: If you will be using Web Services you need to generate ACTDSOAP.cbl and CSTDSOAP.cbl as your driver programs. The sections on Web Service enablement later in this paper will assume that you used these names.

4.1.10 Modifying the CICS converter driver program

You can now modify the converter driver template that was generated by the tool. This is needed to correctly invoke the inbound converter, the existing application, and the outbound converter using the EXEC CICS language. The driver template also provides error-handling mechanisms that can be modified to suit your needs. Modify the mainline section of the ACTDDRV driver program as follows (the necessary changes are highlighted in bold):

```
      *          ******************************************
      *    ********************************************************
      *  **********************************************************
      *               CICS XML Converter Driver Program
      *  **********************************************************
      *    ********************************************************
      *          ******************************************
       Identification Division.
        Program-Id. 'ACTDDRV'.
        Author. WSED.
        Date-Written. 8/7/04 11:27 PM.
      * ************************************************************
      *             Coded Character Sets Configuration
      * ************************************************************
      . . .
      . . .
      * ************************************************************
      *              Business Program Binary Interface
      * ************************************************************
      *01  DFHCOMMAREA .
       01  BUSINESS-STRUCT .
       05  CUSTNO     PIC S99999 .
       05  ACCTNO     PIC S99999 .
       05  BALANCE    PIC S9999V99 .
       Linkage Section.
      * ************************************************************
      *              New Business Program XML Interface
      * ************************************************************
      . . .
      . . .
       Procedure Division using DFHCOMMAREA.
       Mainline Section.
      * ------------------------------------------------------------
      *                   Enable Exception Handler
      * ------------------------------------------------------------
      . . .
      . . .
      * ------------------------------------------------------------
      *               Execute Current Business Program
      * ------------------------------------------------------------
           exec cics link
            program  ('DFH0ACTD')
            commarea (BUSINESS-STRUCT)
           end-exec
      * ------------------------------------------------------------
      *                Execute Outbound XML Converter
      * ------------------------------------------------------------
      . . .
      . . .
      * ------------------------------------------------------------
```

```
      *                              Finished
      * ---------------------------------------------------------------
           exec cics return
           end-exec
           .
       inbound-conversion.
           call 'ACTDCNVI'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
       outbound-conversion.
           call 'ACTDCNVO'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
      . . .
      . . .
       End Program 'ACTDDRV'.
```

Modify the mainline section of the CSTDDRV driver program as follows (the necessary changes are highlighted in bold):

```
      *          ***********************************************
      *    *******************************************************
      *  *********************************************************
      *               CICS XML Converter Driver Program
      *  *********************************************************
      *    *******************************************************
      *          ***********************************************
       Identification Division.
        Program-Id. 'CSTDDRV'.
        Author. WSED.
        Date-Written. 8/7/04 11:27 PM.
      * *************************************************************
      *              Coded Character Sets Configuration
      * *************************************************************
      . . .
      . . .
      * *************************************************************
      *              Business Program Binary Interface
      * *************************************************************
      *01  DFHCOMMAREA .
       01  BUSINESS-STRUCT .
       05  CUSTNO     PIC S99999 .
       05  LASTNAME   PIC A(25) .
       05  FIRSTNAME  PIC A(15) .
       05  ADDRESS1   PIC X(20) .
       05  CITY       PIC A(20) .
       05  STATE      PIC A(10) .
       05  COUNTRY    PIC X(15) .
       Linkage Section.
      * *************************************************************
      *              New Business Program XML Interface
      * *************************************************************
       . . .
       . . .
       Procedure Division using DFHCOMMAREA.
       Mainline Section.
      * -------------------------------------------------------------
      *                   Enable Exception Handler
      * -------------------------------------------------------------
      . . .
      . . .
      * -------------------------------------------------------------
      *               Execute Current Business Program
      * -------------------------------------------------------------
      *    exec cics link
      *     program  ('CURRBUS')
      *     commarea (DFHCOMMAREA)
      *    end-exec

           exec cics link
            program  ('DFH0CSTD')
            commarea (BUSINESS-STRUCT)
           end-exec
      * -------------------------------------------------------------
      *                Execute Outbound XML Converter
```

```
      * -------------------------------------------------------------
      . . .
      . . .
      * -------------------------------------------------------------
      *                            Finished
      * -------------------------------------------------------------
           exec cics return
           end-exec
           .
       inbound-conversion.
           call 'CSTDCNVI'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
       outbound-conversion.
           call 'CSTDCNVO'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
      . . .
      . . .
       End Program 'CSTDDRV'.
```

4.2 Running your XML Enabled application on the mainframe

4.2.1 Preparing your datasets

To set up your XML-enabled application, you allocate the following partitioned data sets then transfer the associated members. The original sources as well as generated programs are shipped in the XML4ESMP.ZIP file in Enterprise Developer. You can use the IDE for z/OS tools to transfer the source files as members to the appropriate datasets on z/OS. Data set characteristics are described in “Setting up and running the existing business application” on page 7.

a. XML.COBOL
- DFH0ACTD*
- DFH0CSTD*
- ACTDCNVI
- ACTDCNVO
- CSTDCNVI
- CSTDCNVO
- ACTDDRV

* These files/steps are not necessary if you installed and ran the existing application described earlier in this paper (See “Setting up and running the existing business application” on page 7).

- CSTDDRV
- XMLFRNT

b. XML.CNTL*
   - DFH$EDB2*
   - DFH$ESQL*
   - XML$CEDA*

c. XML.LOAD*

d. XML.OBJECT*

e. XML.DBRMLIB*

f. XML.BMS*
   - XMLMAP

4.2.2 Configuring DB2*

Use the DB2 sample program DSNTIAD (which is shipped with DB2) to create the DB2 tables "ACCOUNT" and "CUSTOMER" which are needed by the sample programs. Use DFH$EDB2 as a template for creating the tables. Create the tables by running the DB2 sample program DSNTIAD. The DSN SYSTEM (…) parameter is the name of your DB2 subsystem. This DB2 should be connected to your target CICS. You should replace DSTNIAxx with the correct name that corresponds to the release level of your DB2 subsystem, for example, DSTNIA61 for DB2 6.1. Note that starting with release 6 of DB2, DSNTIAD is shipped as source and a load module; in prior releases, it is available as source only. The sample JCL to assemble the DSNTIAD source can be found in DB2.SDSNSAMP(DSNTIJTM) as one of the steps in the control file:

```
//SYSTSIN DD *
      DSN SYSTEM(…)
      RUN PROGRAM(DSNTIAD) PLAN(DSNTIAxx)
      END
//SYSIN  DD    DSN=DFH$EDB2,DISP=SHR
```

4.2.3 Assembling BMS maps *

To create the layout of the CICS front end to the sample programs, you assemble the BMS map XMLMAP using the procedure DFHMAPT supplied with CICS. The resultant COBOL copybook is referenced in the XMLFRNT program.

4.2.4 Pre-compiling the existing programs *

Since the sample programs contain EXEC SQL statements they must be pre-compiled. Use the DB2 pre-compiler DSNHPC to pre-compile the sample programs DFH0ACTD and DFH0CSTD.

4.2.5 Compiling and link-editing the existing application

Compile and Link-Edit the sample and front end programs DFH0ACTD*, DFH0CSTD* and XMLFRNT using the procedure IGYWCL as three separate programs. Be sure to include the CICS compile option. Ensure that the resulting load modules are in a load data set visible to the CICS RPL.

4.2.6 Compiling and link-editing the XML processing code

Compile and Link-Edit the XML Converter and Converter driver programs ACTDDRV, ACTDCNVI, ACTDCNVO, CSTDDRV, CSTDCNVI, CSTDCNVO using the procedure IGYWCL. Link ACTDDRV, ACTDCNVI, ACTDCNVO together as one load module with ACTDDRV as the main program, and CSTDDRV, CSTDCNVI, CSTDCNVO together as one load module with CUSTDDRV as the main program. Please do not specify the CICS compiler option when building these programs as it is provided where appropriate in the source.

* These files/steps are not necessary if you installed and ran the existing application described earlier in this paper (See "Setting up and running the existing business application" on page 7).

4.2.7 Binding the DB2 tables *

To allow the sample programs to access the DB2 tables use the sample DFH$ESQL to perform a DB2 bind for the programs DFH0ACTD and DFH0CSTD. The contents of DFH$ESQL should be as follows:

```
DSN SYSTEM(…)
  BIND PACKAGE(EBUSCOL) -
       OWNER(TONYF) -
       QUALIFIER(TONYF) -
       MEMBER(DFH0CSTD) -
       LIBRARY('TONYF.DBRMLIB.DATA') -
       ACTION(REP) -
       ISOLATION(CS)   -
       VALIDATE(BIND)  -
       DYNAMICRULES(BIND)  -
       ENABLE(CICS)

  BIND PACKAGE(EBUSCOL) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      MEMBER(DFH0ACTD) -
      LIBRARY('TONYF.DBRMLIB.DATA') -
      ACTION(REP) -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)

 BIND PLAN(EBUSPLAN) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      ACTION(REP) -
      PKLIST(EBUSCOL.*)  -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)
   END
 COMMIT;
```

Note that you should use your DB2 subsystem ID as the parameter to the DSN SYSTEM directive. Also replace the highlighted fields with your system's high-level qualifier.

4.2.8 Configuring CICS

Define the various resources to CICS. A sample XML$CEDA is provided to assist with this:

- A transaction named XMLF
- The programs DFH0ACTD*, DFH0CSTD* and XMLFRNT
- The BMS map XMLMAP
- A DB2ENTRY for XMLF that connects it to sample plan EBUSPLAN
- A DB2TRAN for XMLF.

4.2.9 Running the application

To start the application bring up a CICS terminal and run transaction XMLF. The following screen should appear. Note: the following instructions illustrate transaction 1. The procedure for transaction 2 is the same.

```
 _____ _____ _____ _____  __   ____  __ _
/  ____|_   _/  ___|/  ___| \ \ / /  \/  | |
| |      | ||  |     (___     \ V /| \  / | |
| |      | ||  |      \___ \     > < | |\/| | |
| |____ _| ||  |____ ____) |  / . \| |  | | |__
 \_____|_____\_____|_____/  /_/ \_\_|  |_|____|


 PLEASE CHOOSE AN XML ENABLED TRANSACTION:
1. DB2 BANKACCOUNT TABLES FOR THE CUSTOMER DETAILS.
2. DB2 BANKACCOUNT TABLES FOR THE ACCOUNT DETAILS.

  ENTER YOUR CHOICE ====>
```

If you enter 1 as your choice the following will appear on the screen:

```
<?xml version="1.0"?><dfhcommarea>  <custno>1</custno></dfhcommarea>
```

The XML document above represents input to the transaction that will retrieve customer information from a DB2 table. When you press enter the request is executed and the output XML message will appear on the screen that contains the information retrieved from DB2.

```
<cbl:dfhcommarea
xmlns:cbl="http://www.DFH0CSTDO.com/schemas/DFH0CSTDOInterface"><custno>1</cu
stno><lastname>WEAVER</lastname><firstname>RICK</firstname><address1>5 WEST
KIRKWOOD</address1><city>ROANOKE</city><state>TX</state><country>US</country>
</cbl:dfhcommarea>
```

4.2.10 Error reporting

The XML Converters are able to report errors. You can verify that by performing the following test. Clear the terminal screen and run transaction XMLF and select option 1.

When the following screen appears alter one of the XML tags to make the XML document *not well formed* then press enter. For example, change the closing “custno” tag to uppercase “CUSTNO”.

```
<?xml version="1.0"?><dfhcommarea>  <custno>1</custno></dfhcommarea>
```

An XML document containing the error message appears:

```
<errorResponse><errorOccurred>Y</errorOccurred><errorMessageNumber>280</error
MessageNumber><errorCode>005</errorCode><errorCDATAContentLength>230</errorCD
ATAContentLength><errorMessage> <![CDATA[IGZ0280S XML to data structure
conversion could not complete in program "CSTDCNVI" because an error return
code of 5 was received from the XML PARSE statement. The error occurred at
element "custno" with the character content
"1".]]></errorMessage></errorResponse>
```

5 Enabling your business application for Web Services

After creating converters and drivers for XML-enabling your application you may want to take your existing application one step further and expose it as a Web Service. You can use the SOAP for CICS feature at runtime to deliver XML-based SOAP messages to and from your existing business application. The diagram below shows the structure of Web Services-enabled applications that we discussed earlier.

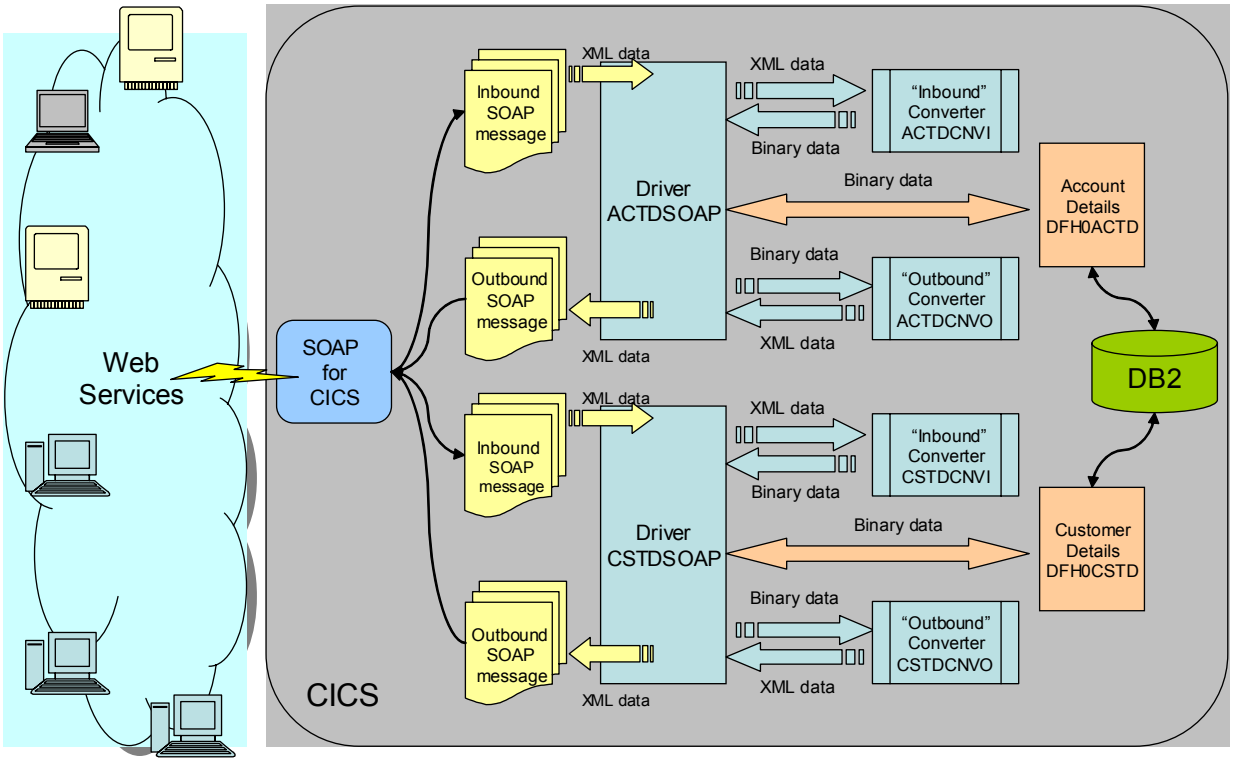


Figure 11, Existing Business CICS Application Enabled for Web Services

The rest of section 5 explains how you can use the Enterprise Developer XML Enablement and Web Services tools to enable and test your application as a Web Service running within the SOAP for CICS environment on z/OS.

5.1 Preparing your application to run as a Web Service on the mainframe

You can now prepare the COBOL converters, drivers and your existing applications to run as Web Services on the mainframe.

5.1.1 Locating your existing application

Locate the sources for DFH0ACTD and DFH0CSTD on your z/OS system. In order to do that, you can use IDE for z/OS tools in Enterprise Developer. You define and connect to the remote z/OS system. For this example, let's assume that the system you connect to is called STLABE1. Once you are connected to STLABE1, locate the sources for DFH0ACTD and DFH0CSTD. Let's assume that they are located in a PDS called XML.COBOL under the high level qualifier (HLQ) TONYF.

5.1.2 Creating an Enterprise Developer project

Create a local container for the generated XML Converters and the Driver template. This local container is called “Simple Project”. In Enterprise Developer, switch to the Resource perspective and invoke the New Project wizard. Select “Simple” as the type of the project you want to create. Follow the wizard to create a project called “XML Account Test” (Fig. 12.)

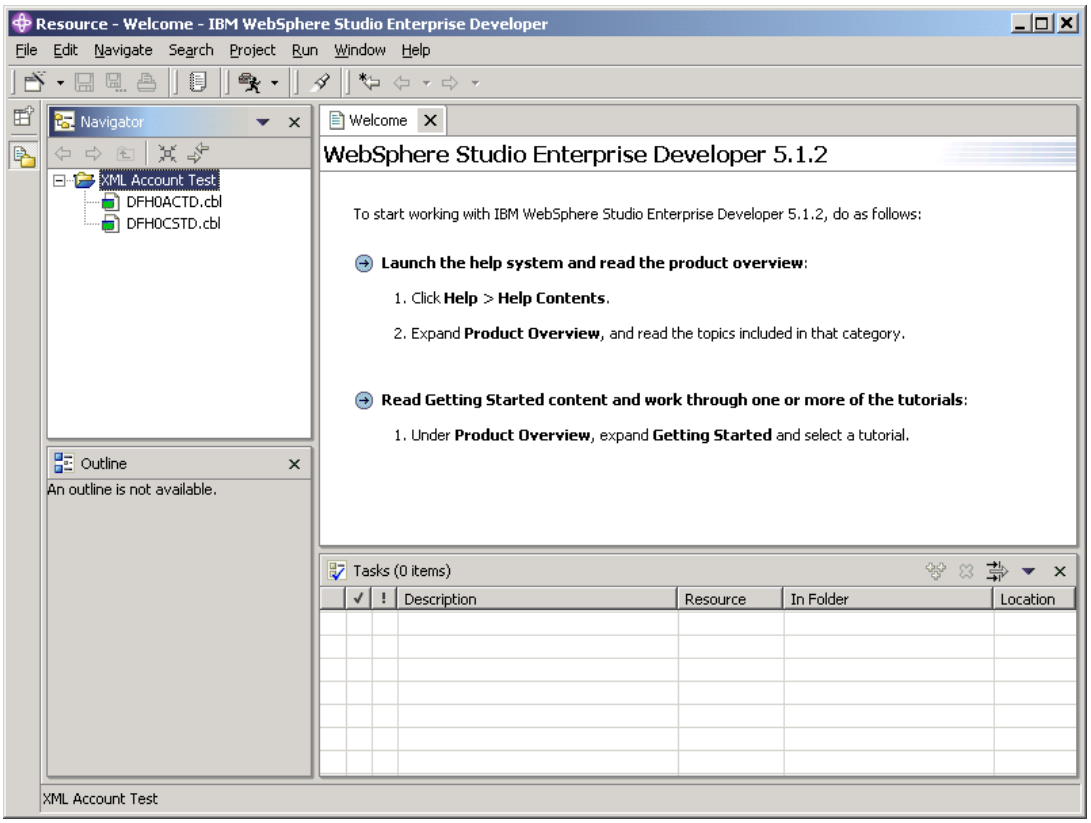


Figure 12, XML Account Test Project

5.1.3 Moving your existing application to Enterprise Developer

Use Copy and Paste operations on the source COBOL files for DFH0ACTD and DFH0CSTD to copy them into your local XML Account Test Project. (Fig. 13) To expand and see the members of the XML.COBOL PDS (which is represented as a Folder in the Resource perspective) you can click the "+".

If the source files are located in the local file system on your workstation, use the File > Import operation provided in the Workbench to import the source files into your local project.

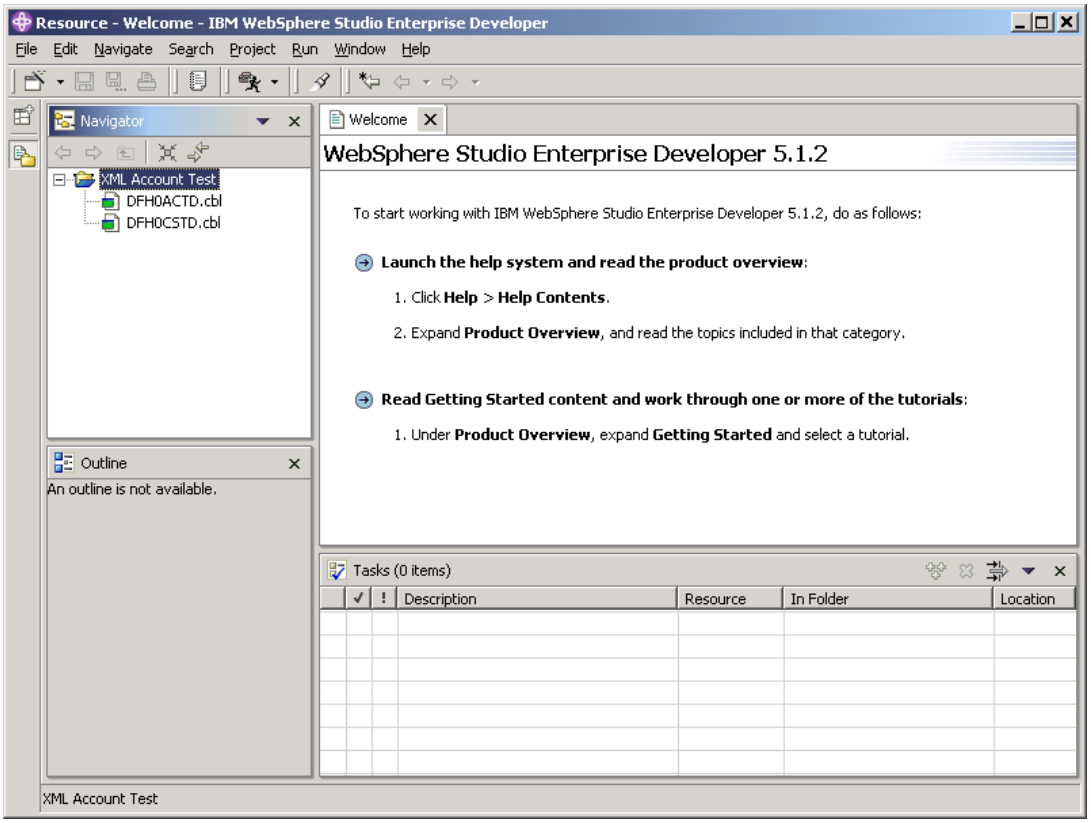


Figure 13, XML Account Test Project With Source

Note that the IDE appends extension `.cbl` to the source file name. This way you can use file type-specific Enterprise Developer tools like a language-sensitive editor.

5.1.4 Invoking the XML Converter generator wizard

Invoke XML Converter generator for the source program DFH0ACTD.cbl. To do that, select DFH0ACTD.cbl in the Navigator view and invoke the pop-up menu by pressing the right mouse button. (Fig. 14)

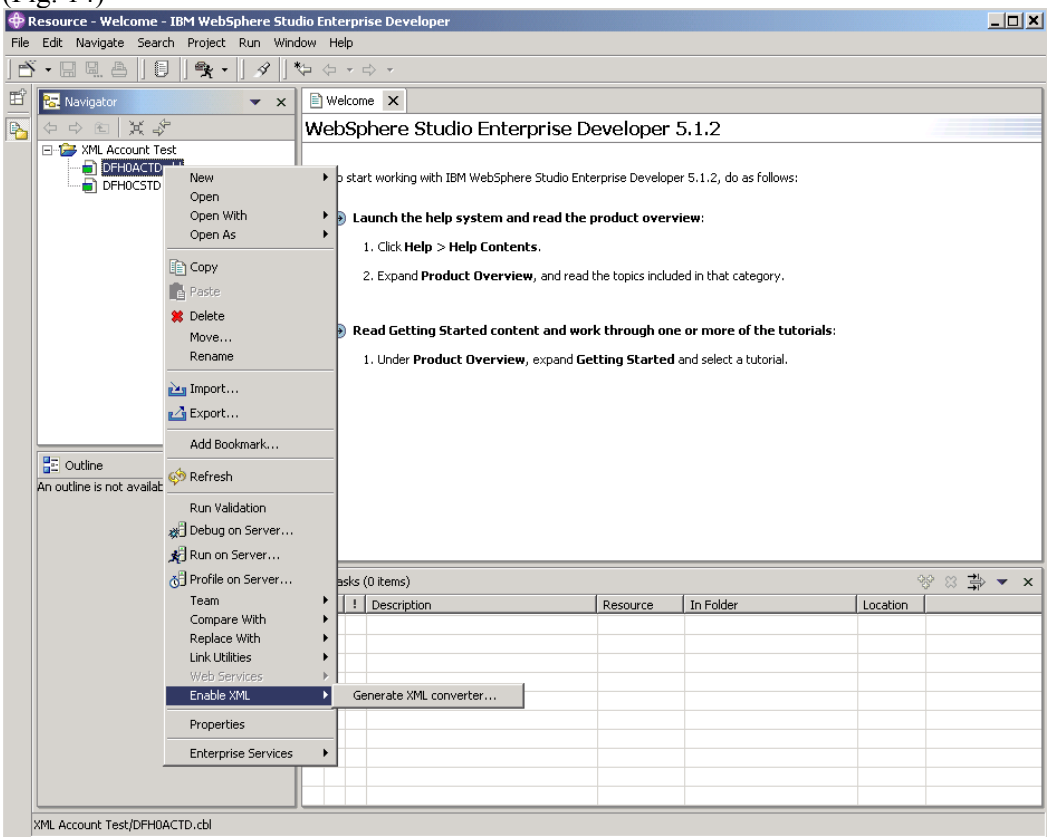


Figure 14, Invoking the XML Converter generator

5.1.5 Specifying input and output files for the wizard

Use the first page of the XML Converter wizard to select input and output files for the Converter, Driver template, and the XML Schema files. Hint: The XML Schema files are automatically generated that contain the description of the names and types of XML elements. These elements can appear in an XML document that our program will process and generate. For more information on XML Schema visit http://www.w3.org/XML/Schema

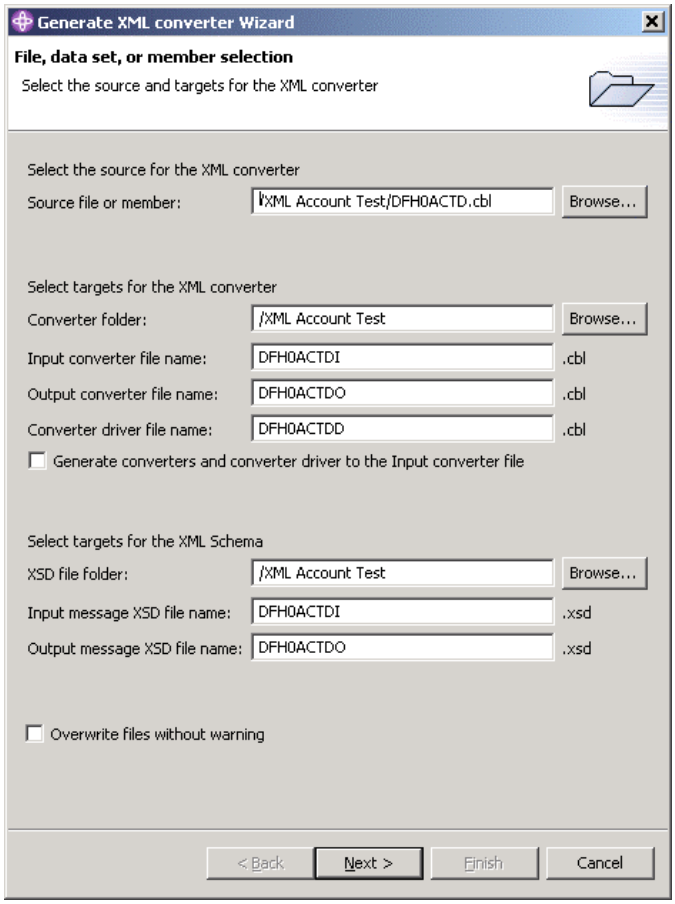


Figure 15, Generate XML Converter Wizard

On this page, the fields are as follows:

- Source file - specify where your existing COBOL program is located
- Converter folders - specify folder(s) where the wizard will generate converter program(s)
- Converter file names - specify the name(s) you want to give your converter file(s),
- Converter driver file name - specify the name you want to give your converter driver.
- Generate converters and converter driver to one file - select if you want to generate the converters and converter driver to the file specified in the "input converter file name" field. The driver will appear first in the file.
- XSD file folders - specify where the wizard will generate the XML Schema files
- XSD file names - specify the names you want to give your XML Schema files
- Overwrite files without warning - select if you want to overwrite existing output files

5.1.6 Specifying the generation options

Use the second page of the XML Converter wizard to specify generation options for your Converter and Driver programs. Choose the host code page for the Host system were you will deploy the converters. Choose "CICS SOAP" as the Driver type. There is no need to modify the rest of the specified defaults on this page. (Fig. 16)

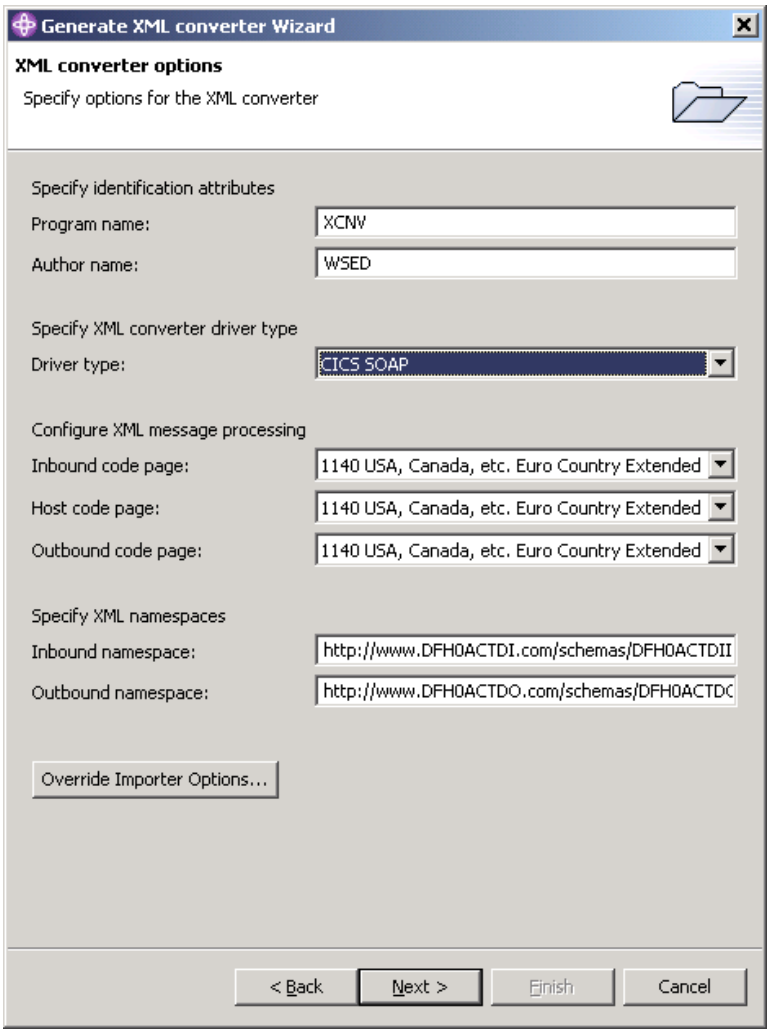


Figure 16, Generation Options Selection

On this page, the fields are as follows:

- Program name – specify the "stem" value for the program names in the PROGRAM-ID paragraph of the IDENTIFICATION DIVISION in the COBOL programs that this wizard will generate. For example, if you enter ACTDCNV, the wizard will generate ACTDCNVI for the inbound converter program name, ACTDCNVO for the outbound converter, and ACTDCNVD for the converter driver.
- Author name – specify the value for the AUTHOR paragraph
- Driver type – specify the desired driver type. Choose "CICS". The CICS driver type provides some conveniences to minimize the need for modifications.
- Maximum message size – specify the maximum size of the XML message that will need to be allocated when processing and generating the XML message.
- Code pages - specify code page(s) for the encoding of the inbound and outbound XML documents, and the code page for the host data.
- Inbound Namespace – specify the inbound namespace or accept the default. This is currently not validated.
- Outbound Namespace – specify the namespace container for messages created by the outbound converter.

Note: The default values for this page are taken from the default preference values stored in the Workbench. You can modify those defaults by visiting the Preference page for the wizard. The process for modifying the preferences is described in “Appendix B. Modifying COBOL Generator preferences” on page 70.

5.1.7 Specifying input and output data structures

Use the third page of the XML Converter wizard to specify the input and output data structures for which you want to generate the equivalent XML-based interface. From the two pull-down combo boxes, select DFHCOMMAREA as both input and output structure. (Fig. 17)

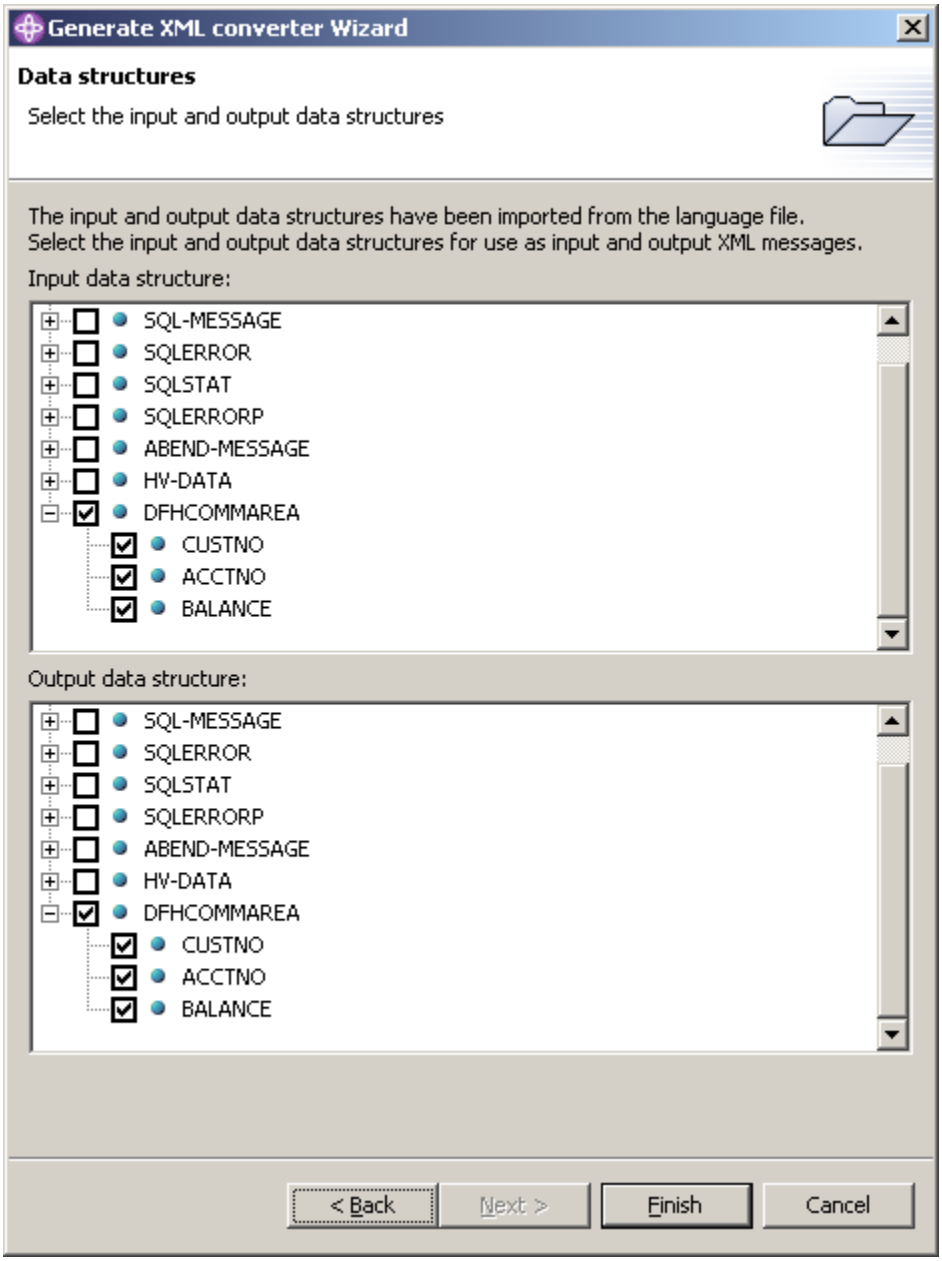


Figure 17, Data Structure Selection

On this page, the fields are as follows:

- `Input data structure` – select the level 01 data from an available list of level 01 COBOL data structures extracted from your existing application program. This data structure will be used to generate inbound XML converters.
- `Output data structure` – same as above but used to generate outbound XML converters.

Note: Some of the attributes for COBOL data items and their treatment by the generated converters depend on COBOL compiler options. An example of such options is the TRUNC(BIN) option that causes the z/OS COBOL compiler to treat all binary items as if they all were native binary items. You can

modify these options using the COBOL Importer preference page. To access this page, click the “override importer options” button on the generation options page in the wizard. The options set during the wizard session do not persist beyond the current session. To permanently modify the importer preferences refer to Appendix C. Modifying COBOL Importer preferences on page 72.

5.1.8 Generating code

Press Finish to complete the generation process. After the wizard processing completes, you will notice that your Converter and Driver files are generated and displayed in the Navigator view. (Fig. 18)

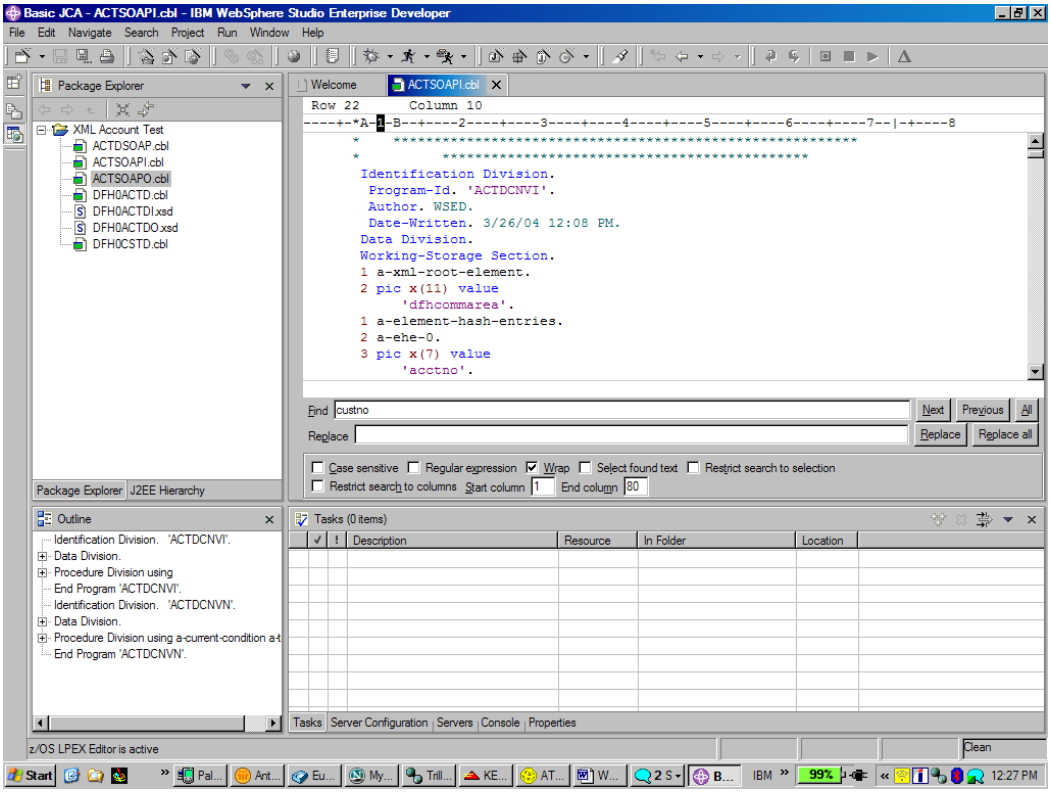


Figure 18, Generated Converters and Driver template for DFH0ACTD

The inbound converter uses the new Enterprise COBOL language, and specifically, the new XML PARSE verb to parse XML documents and convert XML into COBOL data:

```
...
xml parse a-input-xml (1:a-input-xml-len)
       processing procedure a-xml-handler
       thru a-general-logic-exit
      on exception
        perform a-unregister-exception-handler
        perform a-signal-condition
      not on exception
        perform a-unregister-exception-handler
        move zero to a-converter-return-code
  end-xml
...
```

The outbound converter converts the output COBOL data from the existing program into an output XML message:

```
          move CUSTNO
            of DFHCOMMAREA
           to c-CUSTNO-0
            of c-xml-response
```

```
       5 pic x(8) value
           '<custno>'.
       5 c-CUSTNO-0 pic -9(5)
          value zeros.
       5 pic x(9) value
           '</custno>'.
```

5.1.9 Generating additional converters

Use the XML Converter wizard as described in steps above to generate Converters and Driver template for DFH0CSTD. Notice that Converter and Driver files were generated and displayed in the Navigator view. (Fig. 19)

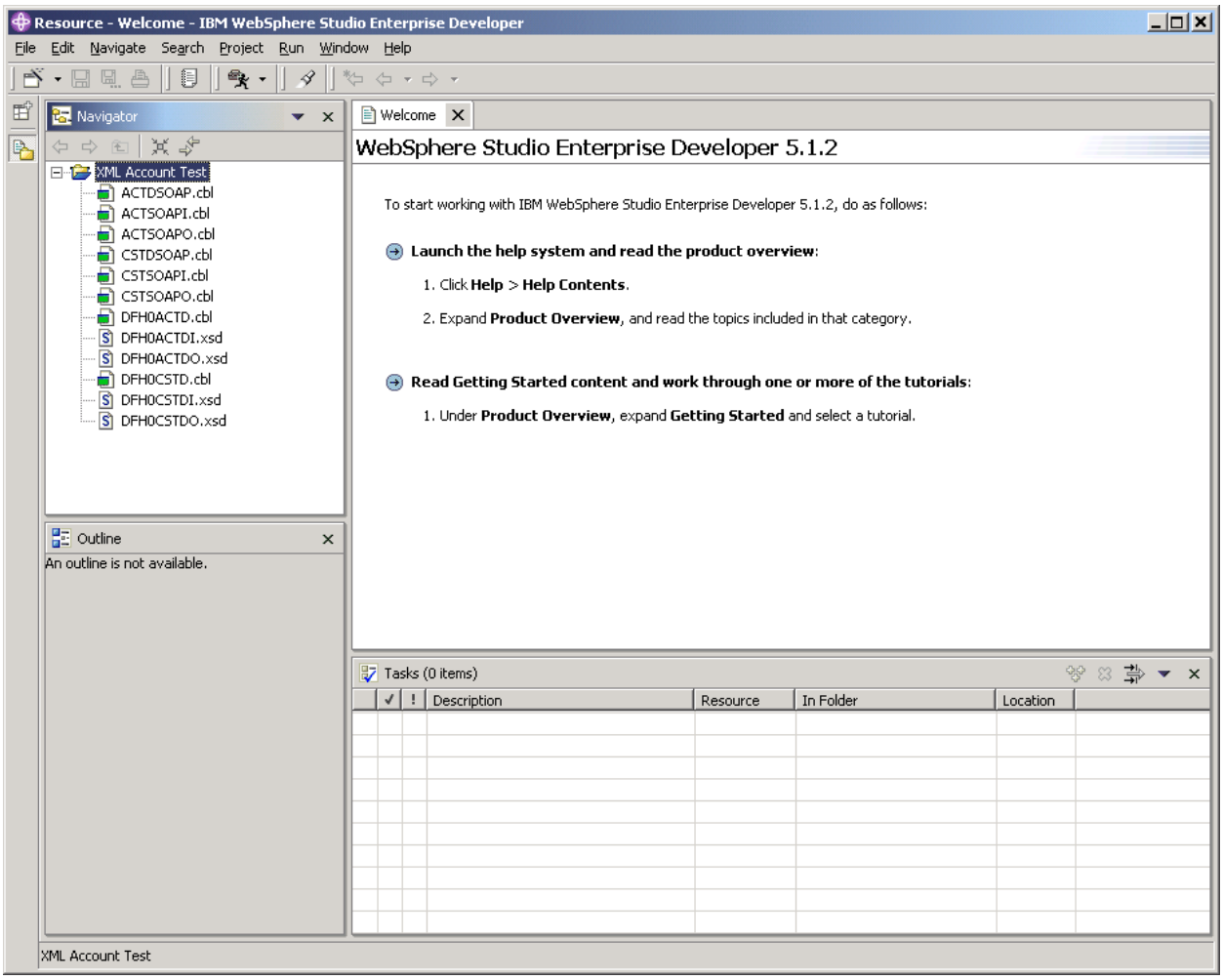


Figure 19, Generated Converters and Driver template for DFH0CSTD

5.1.10 Modifying the CICS SOAP converter driver programs

You can modify the converter driver template that was generated by the tool. This is needed to correctly invoke the inbound converter, the existing application, and the outbound converter using the SOAP for CICS features. The driver template also provides error-handling mechanisms and BTS activity that can be modified to suit your needs. Modify the mainline section of the ACTDSOAP driver program as follows (the necessary changes are highlighted in bold):

```
      *           ***********************************************
      *     *****************************************************
      *   *******************************************************
      *                CICS XML Converter Driver Program
      *   *******************************************************
      *     *****************************************************
      *           ***********************************************
```

```
 Identification Division.
  Program-Id. 'ACTDSOAP'.
  Author. WSED.
  Date-Written. 8/7/04 11:27 PM.
* ************************************************************
*              Coded Character Sets Configuration
* ************************************************************
. . .
. . .
* ************************************************************
*              Business Program Binary Interface
* ************************************************************
*01  DFHCOMMAREA .
 01  BUSINESS-STRUCT .
 05  CUSTNO     PIC S99999 .
 05  ACCTNO     PIC S99999 .
 05  BALANCE    PIC S9999V99 .
* ************************************************************
*                        CICS SOAP BTS
* ************************************************************
. . .
. . .
* ************************************************************
*              New Business Program XML Interface
* ************************************************************
. . .
 PROCEDURE DIVISION.
 MAINLINE SECTION.
* ------------------------------------------------------------
*                   Enable Exception Handler
* ------------------------------------------------------------
. . .
. . .
* ------------------------------------------------------------
*                Execute Inbound XML Converter
* ------------------------------------------------------------
. . .
* ------------------------------------------------------------
*               Execute Current Business Program
* ------------------------------------------------------------
*    exec cics link
*     program  ('CURRBUS')
*     commarea (DFHCOMMAREA)
*    end-exec

     exec cics link
      program  ('DFH0ACTD')
      commarea (BUSINESS-STRUCT)
     end-exec
* ------------------------------------------------------------
*                Execute Outbound XML Converter
* ------------------------------------------------------------
. . .
. . .
* ------------------------------------------------------------
*                           Finished
* ------------------------------------------------------------
     exec cics return
     end-exec
```

```
           .
       inbound-conversion.
           call 'ACTSOAPI'
             using
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
       outbound-conversion.
           call 'ACTSOAPO'
             using
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
      . . .
      . . .
       End Program 'ACTDSOAP'.
```

Modify the mainline section of the CSTDSOAP driver program as follows (the necessary changes are highlighted in bold):

```
      *          ******************************************
      *    ********************************************************
      *  **********************************************************
      *               CICS XML Converter Driver Program
      *  **********************************************************
      *    ********************************************************
      *          ******************************************
       Identification Division.
        Program-Id. 'CSTDSOAP'.
        Author. WSED.
        Date-Written. 8/7/04 11:27 PM.
      * ************************************************************
      *              Coded Character Sets Configuration
      * ************************************************************
      . . .
      . . .
      * ************************************************************
      *              Business Program Binary Interface
      * ************************************************************
      *01  DFHCOMMAREA .
       01  BUSINESS-STRUCT .
       05  CUSTNO     PIC S99999 .
       05  LASTNAME   PIC A(25) .
       05  FIRSTNAME  PIC A(15) .
       05  ADDRESS1   PIC X(20) .
       05  CITY       PIC A(20) .
       05  STATE      PIC A(10) .
       05  COUNTRY    PIC X(15) .
      * ************************************************************
      *                         CICS SOAP BTS
      * ************************************************************
      . . .
      . . .
      * ************************************************************
      *              New Business Program XML Interface
      * ************************************************************
      . . .
      . . .
       Procedure Division.
       Mainline Section.
      * ------------------------------------------------------------
      *                   Enable Exception Handler
      * ------------------------------------------------------------
      . . .
      . . .
      * ------------------------------------------------------------
      *               Execute Current Business Program
      * ------------------------------------------------------------

           exec cics link
            program  ('DFH0CSTD')
            commarea (BUSINESS-STRUCT)
           end-exec
```

```
      * -------------------------------------------------------------
      *                Execute Outbound XML Converter
      * -------------------------------------------------------------
      . . .
      . . .
      * -------------------------------------------------------------
      *                           Finished
      * -------------------------------------------------------------
           exec cics return
           end-exec
           .
       inbound-conversion.
           call 'CSTSOAPI'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
       outbound-conversion.
           call 'CSTSOAPO'
             using
      *        DFHCOMMAREA
               BUSINESS-STRUCT
               xml-int-len
               xml-int-txt
               omitted
      *    optional-feedback-code
             returning
               converter-return-code
           .
      . . .
      . . .
       End Program 'CSTDSOAP'.
```

5.1.11 Preparing your datasets

To set up your Web-enabled application, you allocate the following partitioned data sets then transfer the associated members. The original sources as well as generated programs are shipped in the XML4ESMP.ZIP file in Enterprise Developer. You can use the IDE for z/OS tools to transfer the source files as members to the appropriate datasets on z/OS. Data set characteristics are described in “Setting up and running the existing business application” on page 7.

a. XML.COBOL

- DFH0ACTD*
- DFH0CSTD*
- ACTSOAPI
- ACTSOAPO
- CSTSOAPI
- CSTSOAPO
- ACTDSOAP

* These files/steps are not necessary if you installed and ran the existing application described earlier in this paper (See “Setting up and running the existing business application” on page 7).

- CSTDSOAP

b. XML.CNTL*
   - DFH$EDB2*
   - DFH$ESQL*
   - XML$CEDA*

c. XML.LOAD*

d. XML.OBJECT*

e. XML.DBRMLIB*

f. XML.BMS*
   - XMLMAP

5.1.12 Configuring DB2*

Use the DB2 sample program DSNTIAD (which is shipped with DB2) to create the DB2 tables "ACCOUNT" and "CUSTOMER" which are needed by the sample programs. Use DFH$EDB2 as a template for creating the tables. Create the tables by running the DB2 sample program DSNTIAD. The DSN SYSTEM (…) parameter is the name of your DB2 subsystem. This DB2 should be connected to your target CICS. You should replace DSTNIAxx with the correct name that corresponds to the release level of your DB2 subsystem, for example, DSTNIA61 for DB2 6.1. Note that starting with release 6 of DB2, DSNTIAD is shipped as source and a load module; in prior releases, it is available as source only. The sample JCL to assemble the DSNTIAD source can be found in DB2.SDSNSAMP(DSNTIJTM) as one of the steps in the control file:

```
//SYSTSIN DD *
      DSN SYSTEM(…)
      RUN PROGRAM(DSNTIAD) PLAN(DSNTIAxx)
      END
//SYSIN  DD    DSN=DFH$EDB2,DISP=SHR
```

5.1.13 Pre-compiling the existing programs*

Since the sample programs contain EXEC SQL statements they must be pre-compiled. Use the DB2 pre-compiler DSNHPC to pre-compile the sample programs DFH0ACTD and DFH0CSTD.

5.1.14 Compiling and link-editing the existing application*

Compile and Link-Edit the sample and front end programs DFH0ACTD*, DFH0CSTD* and XMLFRNT* using the procedure IGYWCL. Be sure to include the CICS compile option. Ensure that the resulting load modules are in a load data set visible to the CICS RPL.

5.1.15 Compiling and link-editing the XML processing code

Compile and Link-Edit the XML Converter and Converter driver programs ACTDSOAP, ACTDCNVI, ACTDCNVO, CSTDSOAP, CSTDCNVI, CSTDCNVO using the procedure IGYWCL. Link ACTDSOAP, ACTDCNVI, ACTDCNVO together as one load module with ACTDSOAP as the main program, and CSTDSOAP, CSTDCNVI, CSTDCNVO together as one load module with CSTDSOAP as the main program. Please do not specify the CICS compiler option when building these programs as it is provided where appropriate in the source.

* These files/steps are not necessary if you installed and ran the existing application described earlier in this paper (See "Setting up and running the existing business application" on page 7).

5.1.16 Binding the DB2 tables *

To allow the sample programs to access the DB2 tables use the sample DFH$ESQL to perform a DB2 bind for the programs DFH0ACTD and DFH0CSTD. The contents of DFH$ESQL should be as follows:

```
DSN SYSTEM(…)
  BIND PACKAGE(EBUSCOL) -
       OWNER(TONYF) -
       QUALIFIER(TONYF) -
       MEMBER(DFH0CSTD) -
       LIBRARY('TONYF.DBRMLIB.DATA') -
       ACTION(REP) -
       ISOLATION(CS)   -
       VALIDATE(BIND)  -
       DYNAMICRULES(BIND)  -
       ENABLE(CICS)

  BIND PACKAGE(EBUSCOL) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      MEMBER(DFH0ACTD) -
      LIBRARY('TONYF.DBRMLIB.DATA') -
      ACTION(REP) -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)

 BIND PLAN(EBUSPLAN) -
      OWNER(TONYF) -
      QUALIFIER(TONYF) -
      ACTION(REP) -
      PKLIST(EBUSCOL.*)  -
      ISOLATION(CS)   -
      VALIDATE(BIND)  -
      DYNAMICRULES(BIND)  -
      ENABLE(CICS)
   END
 COMMIT;
```

Note that you should use your DB2 subsystem ID as the parameter to the DSN SYSTEM directive. Also replace the highlighted fields with your system's high-level qualifier.

5.1.17 Configuring CICS

Define the various resources to CICS. A sample XML$CEDA is provided to assist with this:

- A transaction named XMLS
- The programs DFH0ACTD*, DFH0CSTD*, CUSTDSOAP and ACTDSOAP
- A DB2ENTRY for XMLF* that connects it to sample plan EBUSPLAN
- A DB2TRAN for XMLS.

5.2 Creating a SOAP-Based Web Service client for an Enterprise Application

In the previous sections, you created XML converters and an XML Schemas (XSD) for inbound and outbound messages (for example, DFH0ACTDI.xsd for inbound and DFH0ACTDO.xsd for outbound).

Now, let's create a Web Services description for the Account details service. This service is already implemented! Surprised? Don't be! Your XML enabled business application described in the previous sections is the implementation of your web service. The only thing left to do is to describe that service to the Web services world.

WSDL or Web Service Description language is an XML format for describing network services. It provides a description of the service, consumable in many other technologies (such as in Enterprise Developer to generate Java[1] code to drive the service) or UDDI to publish the service to other organizations.

The definition includes the thought of considering services as a set of endpoints operating on messages. The operations and/or procedures contained in these services are first described in an abstract manner, then bound to a specific network, transport protocol and message format.

WSDL is based on other technologies such as XML schemas and XML namespaces. WSDL provides a description not only of the service, but also of the messages the service can accept and generate.

You will use the Web Services New WSDL wizard. To run it, select File > New > Other > Web Services > WSDL. The following Wizard appears:

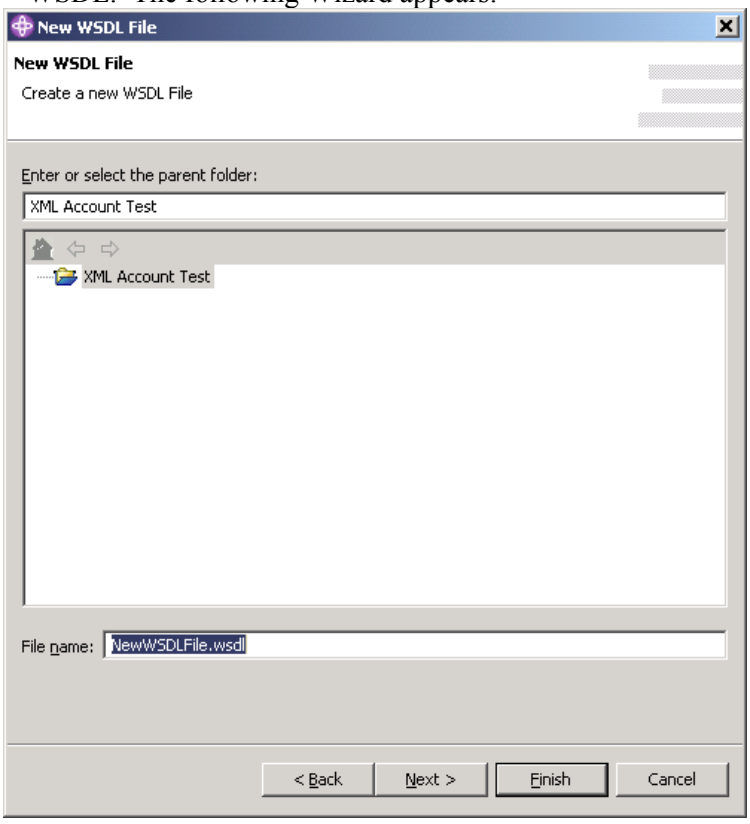


Figure 20, New WSDL File Wizard

Name your WSDL (For example, DFH0ACTD.wsdl), click Next and the following page appears:

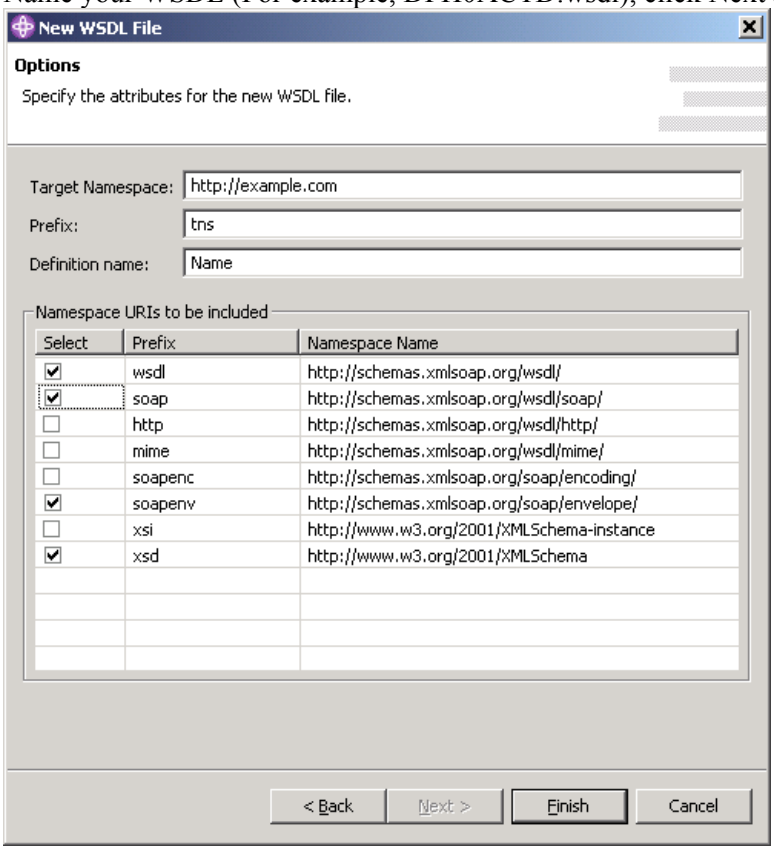


Figure 21, Specifying New WSDL File Options

You can also give your WSDL a name; for example, AcctDetailWsdl.

Namespaces are external locations that contain definitions for various parts of the WSDL we are defining – including the elements, attributes, and types. These definitions define various parameters including SOAP (protocol) and Envelope (data or payload).

Type the name into the Definition Name field. Make sure to select **soap** and **soapenv** namespace URIs, and click Finish.

The new WSDL file is created and added to your project:

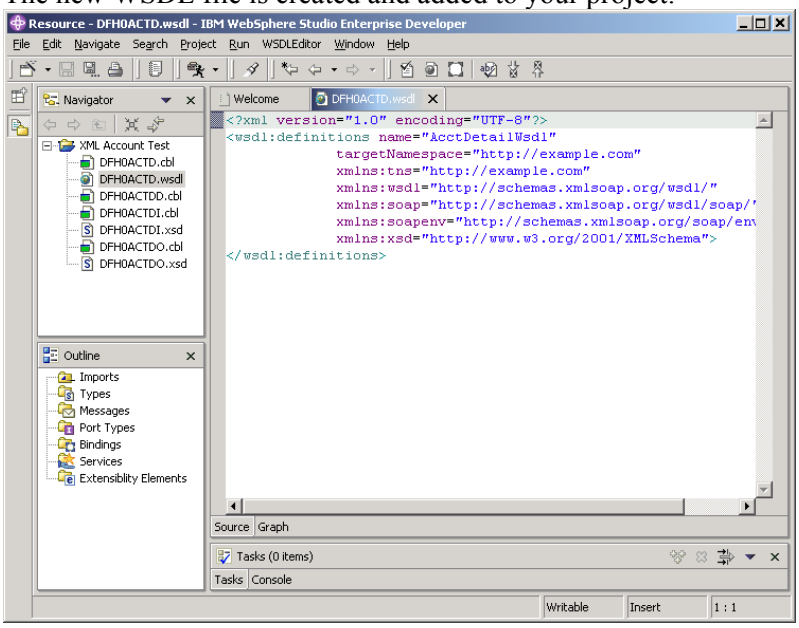


Figure 22, New WSDL file

Now switch to the "Graph" view of the WSDL source.

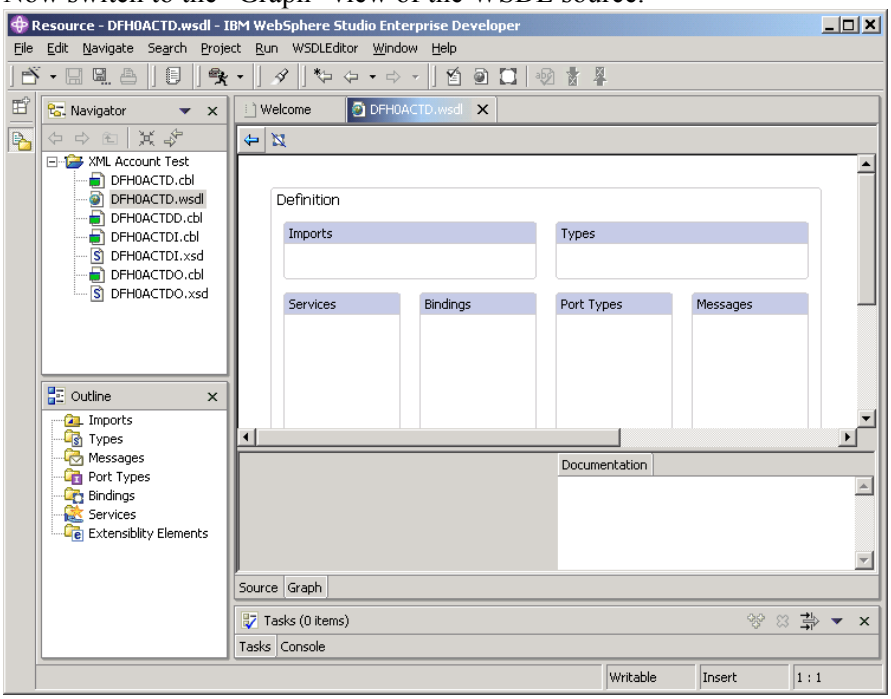


Figure 23, Graph View of WSDL Source

Fill in the additional details in WSDL definition needed to complete the service.

Above we discussed web services and messages contained therein. Messages are logically grouped to operations that are logically grouped to port types. You'll start the process then by defining a Port Type.

Create a Port Type for your service: Right-click on the "Port Type" section in the Graph view, then select Add Child > portType. The following dialog appears:

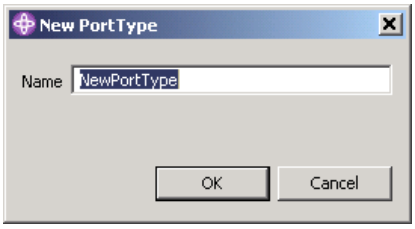


Figure 24, New Port Type Dialog

Type your port type name in this dialog (for example, AcctDetailPortType) and click OK. AcctDetailPortType entry is created in the Port Types section.

Now create an Operation for the Port Type. Operations represent a unit of work and also can be looked on as representing an endpoint, such as our CICS program(s) or service(s). Each operation is composed of messages, for example input and output messages.

Right-click on AcctDetailPortType then select Add child > operation. The following dialog appears:

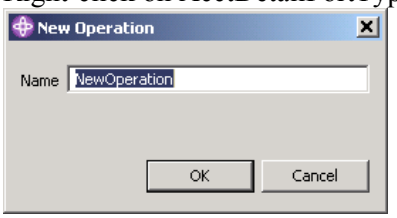


Figure 25, New Operation Dialog

Type your operation name in this dialog (for example, AcctDetailOperation) and click OK. AcctDetailOperation entry is created under AcctDetailPortType in the Port Types section:

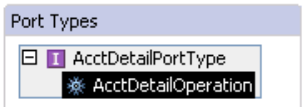


Figure 26, Port Types Section

Now, in the Messages section, create a message to be used with your AcctDetailOperation. The message will have two parts. A request part (or input) and a response part (or output). Right mouse click on the "Messages" section in the Graph view then select Add Child > message. The following dialog appears:

Figure 27, New Message Dialog

Type your message name in this dialog (for example, AcctDetailMessage) and click OK. AcctDetailMessage is created in the Messages section.

Now create message parts. Right mouse click on AcctDetailMessage then select Add child > part. The following dialog appears:

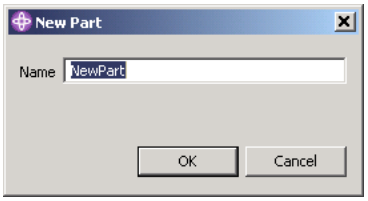


Figure 28, New Part Dialog

Type your part name in this dialog (for example, AcctDetailRequest) and click OK.
AcctDetailRequest part is created under AcctDetailMessage in the Messages section:

Figure 29, Messages Section

Note that the part is initially created with a type of xsd:string. We will change that a bit later.
Next create a AcctDetailResponse part for your message the same way as you created AcctDetailRequest:

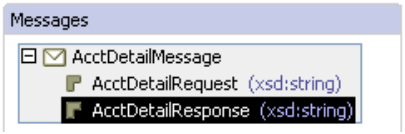


Figure 30, Messages Section with Additional Part

Next, you will need to define the message parts according to the XML Schema types derived from the COBOL data structures. Use the XML Schema(s) or XSD(s) that define the message (inbound and outbound). In some cases the schema will be the same, in some it will differ between input and output.
To do that, in the Message section, right mouse click on the AcctDetailRequest. Then in the action menu, click Set Element… action. The following dialog appears:

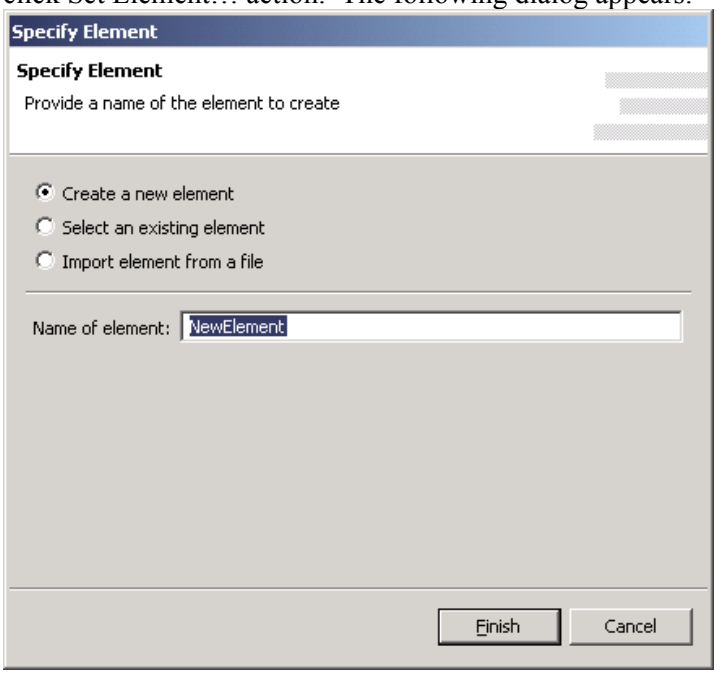


Figure 31, Specify Element Dialog

Select the Import element from a file radio button.
The dialog will change to the following:

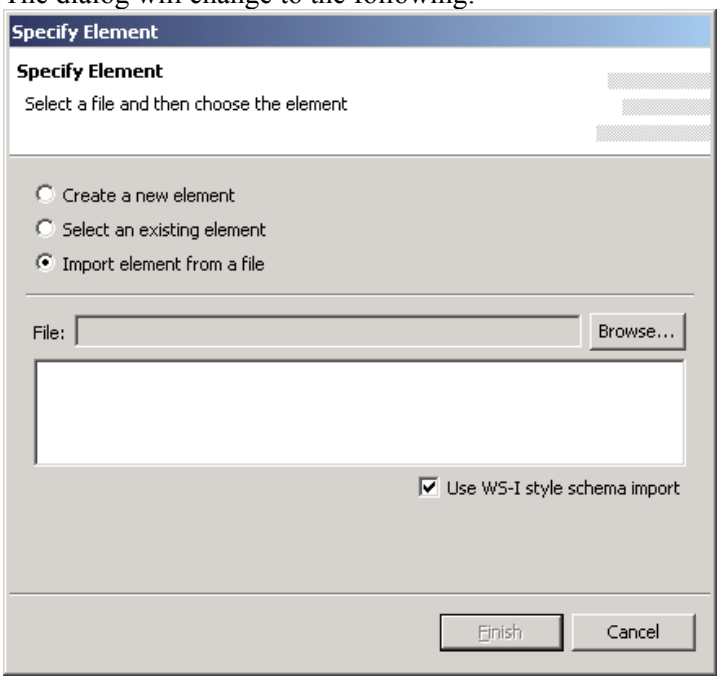


Figure 32, Specify Element Dialog with Import Option Selected

Click Browse.

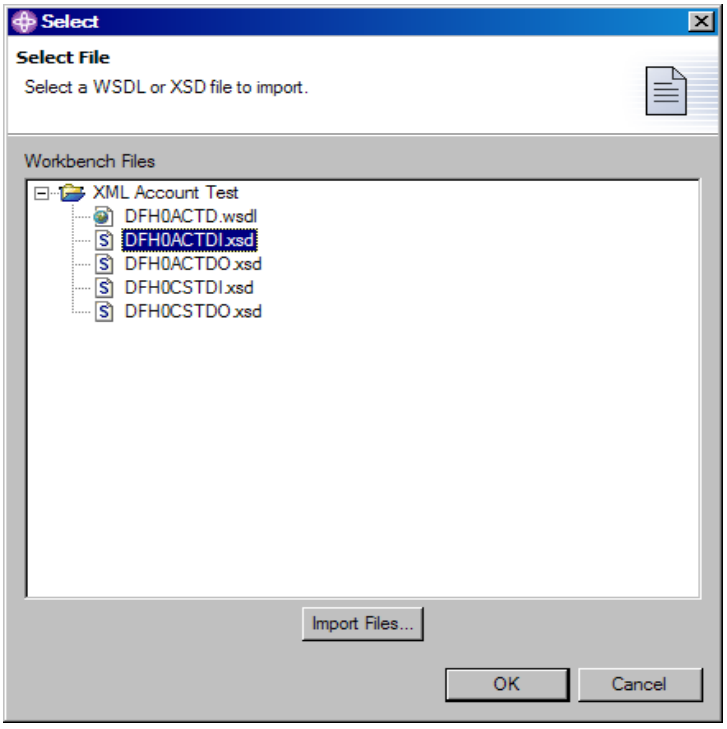


Figure 33, Importing an Element

In the File selection dialog that follows, navigate to the XSD file that was produced from the COBOL inbound structure, for example DFH0ACTDI.xsd.

Click OK in the File selection dialog and the following will appear in the Type selection:

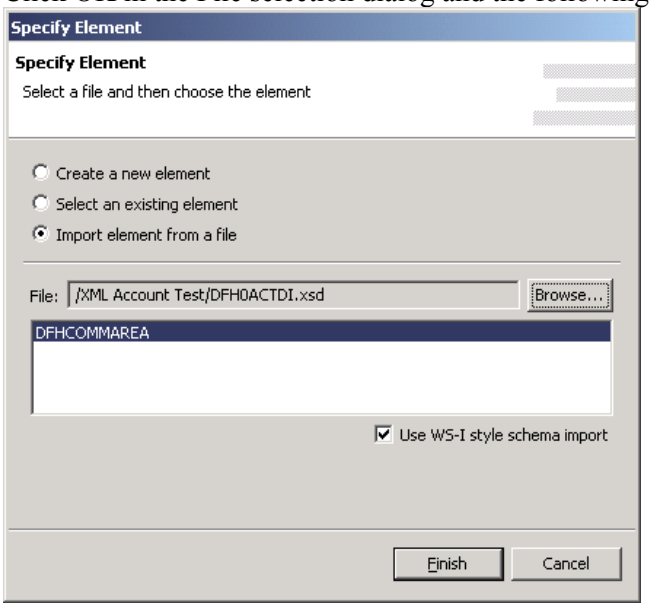


Figure 34, Completing Element Specification

Select DFHCOMMAREA. Also make sure that **Use WS-I style schema import** is checked. Click Finish.

Use the same process to set the response part element, but when selecting the XSD file, choose the outbound XSD, for example, DFH0ACTDO.xsd.
At the end of specifying part elements, the Messages section will look like this:

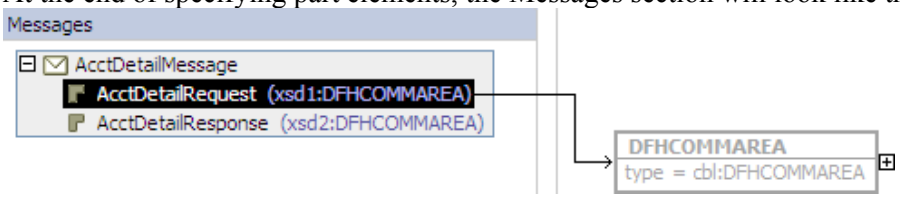


Figure 35, Messages Section Showing an External Element Reference

Note that the shadowed rectangle pointed to by the arrow indicates an external element reference.

Up until this point, everything you've done logically defines the service. Now you will begin the process of binding your operation to the protocol and to the physical endpoints.

A port ties your message to a protocol-specific address and/or network endpoints.

Now create input and output parameters for your operation. In the Port Types section, right mouse click on AcctDetailOperation then select Add child > input. Do the same for output. "input" and "output" parameters are created under AcctDetailOperation in the Port Types section:

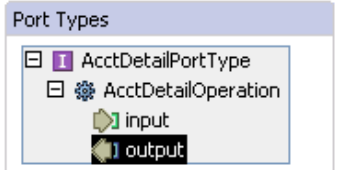


Figure 36, Port Types Section Showing Input and Output Parameters

You will associate a message from the Messages section with the input and output parameters for your operation. You can use either the context menu action, Set message, or the Input editor section. Let's use the Input editor:

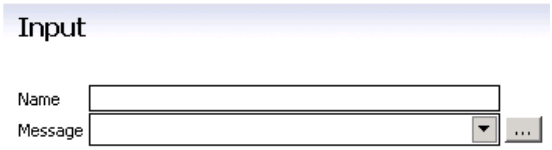


Figure 37, Input Editor

In the Name entry enter, for example, AcctDetailOperationInput.

In the Message entry, click on the combo box arrow pull-down and select the Message that we created earlier:

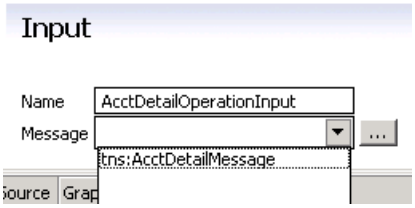


Figure 38, Input Editor Message Selection

Repeat the same for the output parameter, but enter for example, AcctDetailOperationOutput as output parameter name.

Next you will create a binding for your Port Type. This completes the protocol/endpoint definition mapping the logical service to the physical endpoints.

To create a binding right mouse click on the Binding section of the WSDL editor and select Add Child->Binding:

Figure 39, Binding Wizard Toolbar Icon

The Binding wizard appears:

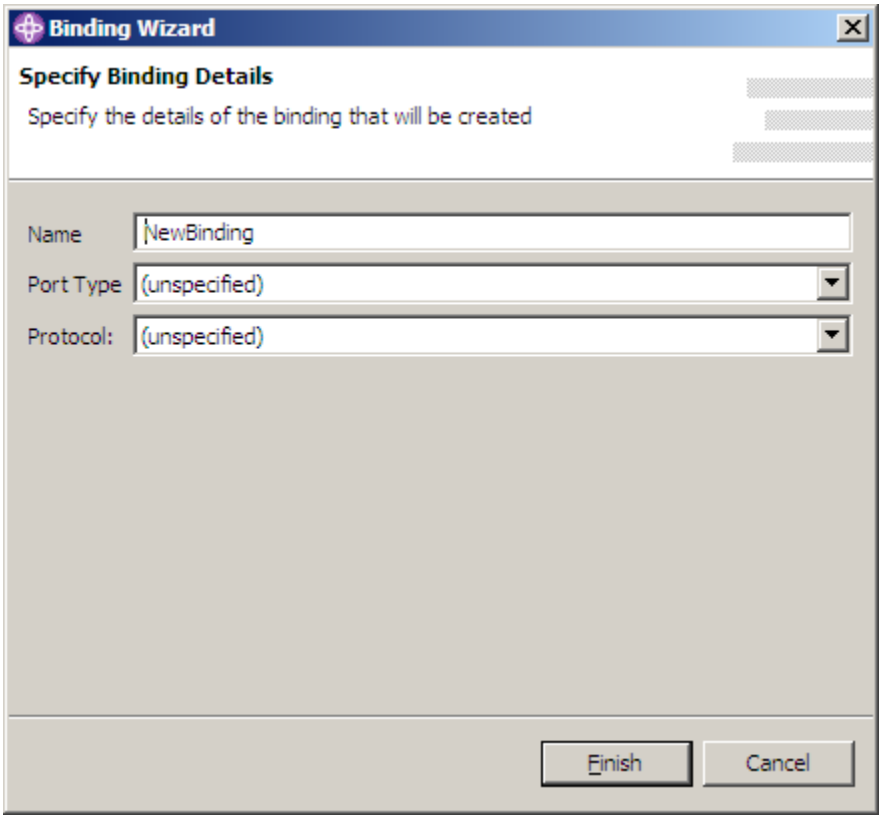


Figure 40, Generate Binding Wizard

Enter a name for your binding, for example, CICSSOAPBinding. In the Port Type combo box arrow pull-down, select the existing port type created earlier.
In the Protocol combo box arrow pull-down, select the SOAP protocol, and make sure that **document literal** is checked as SOAP Binding Options:

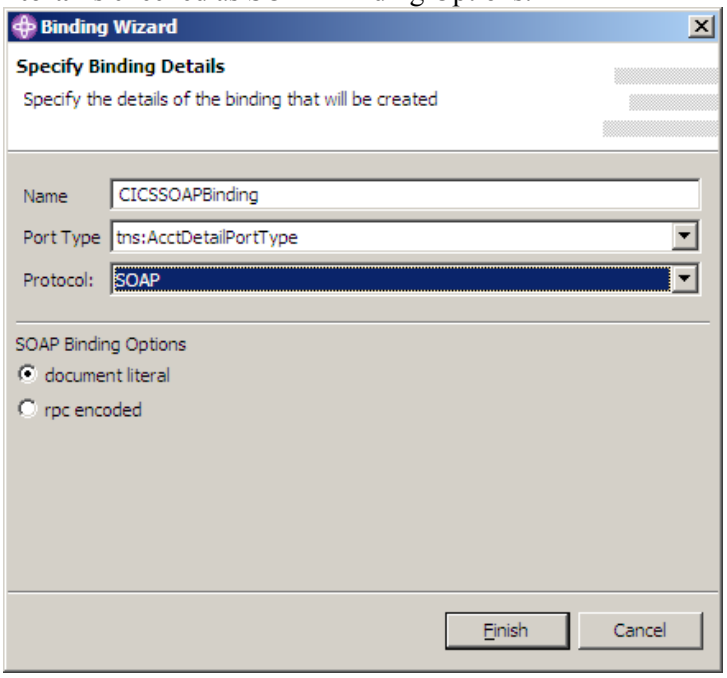


Figure 41, SOAP Binding Options in Generate Binding Wizard

Click Finish. The Binding is now linked with Port Type(s).

Your new Binding will appear in the Graph view:

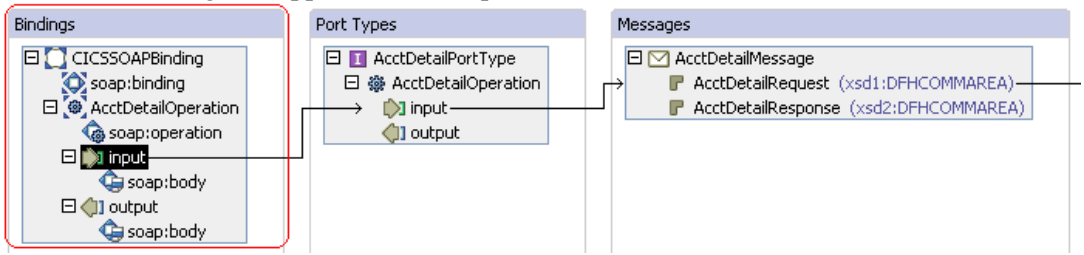


Figure 42, Graph View Showing New Binding

You will need to set the **soap:binding** and **soap:operation** styles to **document.**
This is done using the soap:binding and soap:operation editor sections. Bring up the respective editor sections by clicking on soap:binding and soap:operation under Bindings.

Figure 43, Setting Binding and Operation Styles

The style entry is a pull-down combo box. Make sure that you select document as style for both

soap:binding and soap:operation.

You'll finish the binding by associating input **soap:body** with the **AcctDetailRequest** part and output **soap:body** with the **AcctDetailResponse** part. You do that using the soap:body editor sections. Type **AcctDetailRequest** and **AcctDetailResponse** into the "parts" property:

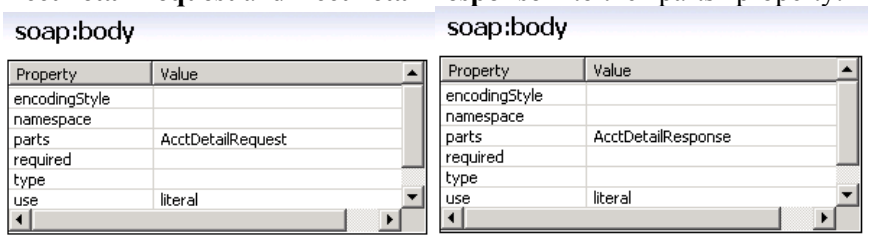


Figure 44, Completing the Binding

Add the CICS SOAP Service. The service definition completes our task by associating an address or endpoint to the binding.

Using the Services section you will create the service. Then right-click on the Services section heading. Select Add child > service. The following dialog appears:

Figure 45, Creating the Service

Type your service name in this dialog (for example, CICSSOAPService) and click OK. CICSSOAPService service is created in the Services section:

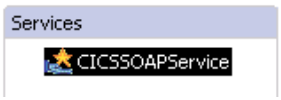


Figure 46, Services Section

Now create a port for your CICSSOAPService. In the Services section of the graph view, right mouse click on CICSSOAPService then select Add Child->Port. In the ensuing dialog, type your port name in the Name field (for example, CICSSOAPServicePort), select CICSSOAPBinding as Binding and SOAP as protocol. In the SOAP Port details' Location field, type the HTTP address of the CICS SOAP server you wish to access with this Web Service, for example,
http://winmvsn0.cpit.hursley.ibm.com:8888/CICS/XMLS/DFHWSDSH/ACTDSOAP
(The URL is used to communicate with the back end server and server side program(s) and it represents the CICS SOAP XMLE driver on the z/OS system.)
The filled Port dialog is shown in Fig. 47.

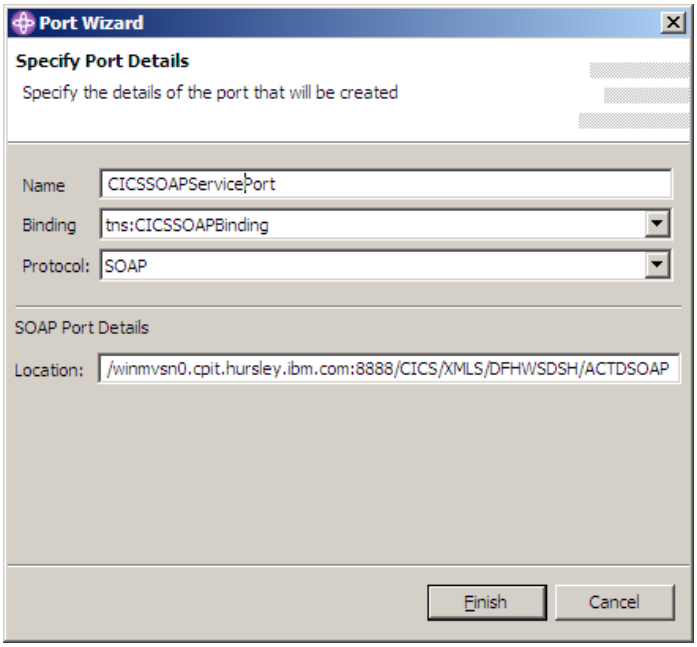


Figure 47, Creating a Port

Your WSDL file is complete. You can save the WSDL file and verify that it is correct by running the WSDL verification. Select the WSDL verification tool icon on the tool bar:

Figure 48, Validate WSDL Document Icon

If the WSDL is correct, you will see the following message:

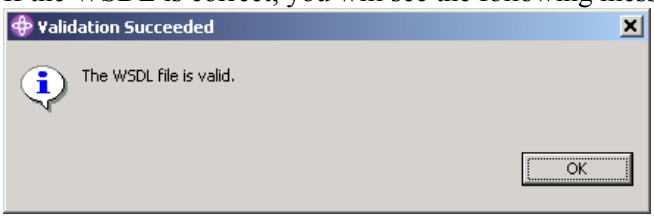


Figure 49, Validation Succeeded Dialog

The final WSDL file is shown here:

```
 <?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="AcctDetailWsdl"
             targetNamespace="http://example.com"
             xmlns:tns="http://example.com"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsd1="http://www.DFH0ACTDI.com/schemas/DFH0ACTDIInterface"
xmlns:xsd2="http://www.DFH0ACTDO.com/schemas/DFH0ACTDOInterface">
        <wsdl:types>
                <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                        <xsd:import
namespace="http://www.DFH0ACTDI.com/schemas/DFH0ACTDIInterface"
schemaLocation="DFH0ACTDI.xsd"></xsd:import>
                        <xsd:import
namespace="http://www.DFH0ACTDO.com/schemas/DFH0ACTDOInterface"
schemaLocation="DFH0ACTDO.xsd"></xsd:import>
                </xsd:schema>
        </wsdl:types>
        <wsdl:message name="AcctDetailMessage">
```

```
                <wsdl:part name="AcctDetailRequest" element="xsd1:DFHCOMMAREA"></wsdl:part>
                <wsdl:part name="AcctDetailResponse" element="xsd2:DFHCOMMAREA"></wsdl:part>
        </wsdl:message>
        <wsdl:portType name="AcctDetailPortType">
                <wsdl:operation name="AcctDetailOperation">
                        <wsdl:input message="tns:AcctDetailMessage"
name="AcctDetailOperationInput"></wsdl:input>
                        <wsdl:output message="tns:AcctDetailMessage"
name="AcctDetailOperationOutput "></wsdl:output>
                </wsdl:operation>
        </wsdl:portType>
        <wsdl:binding name="CICSSOAPBinding" type="tns:AcctDetailPortType">
                <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
                <wsdl:operation name="AcctDetailOperation">
                        <soap:operation soapAction="http://example.com/AcctDetailOperation"
style="document"/>
                        <wsdl:input name="AcctDetailOperationInput">
                                <soap:body use="literal" parts="AcctDetailRequest"/>
                        </wsdl:input>
                        <wsdl:output name="AcctDetailOperationOutput ">
                                <soap:body use="literal" parts="AcctDetailResponse"/>
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="CICSSOAPService">
                <wsdl:port name="CICSSOAPServicePort" binding="tns:CICSSOAPBinding">
                        <soap:address
location="http://winmvsn0.cpit.hursley.ibm.com:8888/CICS/XMLS/DFHWSDSH/ACTDSOAP"/>
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

5.3 Using the Batch Processor (XSEBATCH)

New in WSED Version 5.1.2 is the ability to create the complete set of converters, drivers, schemas, and WSDL files using a command-line batch program. The use of this program, XSEBATCH, is described in detail (complete with samples) in the on-line help for WSED 5.1.2. To invoke on-line help in WSED, select Help->Help Contents in the workbench toolbar. When the help is displayed, navigate to the Batch Process chapter of the Developing section:

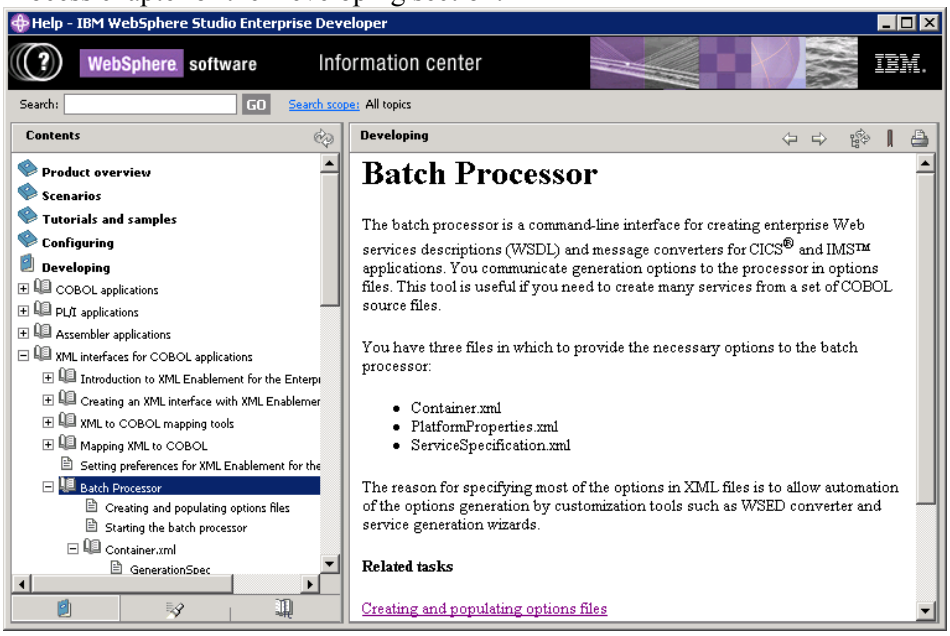


Figure 50, Batch processor on-line help

If you intend to create web services from many business applications at once, the use of the batch processor greatly simplifies generation of multiple sets of converters and the WSDL files.

5.4 Testing your Web Services-enabled business application

To test your Web Services enabled application you will use Web Services explorer. You can find the complete set of documentation on using the Web Services Explorer in the on-line WebSphere Studio Information Center.

Right-click on the WSDL file you created and select Web Services > Test with Web Services Explorer. You will see the Web Services Explorer view:

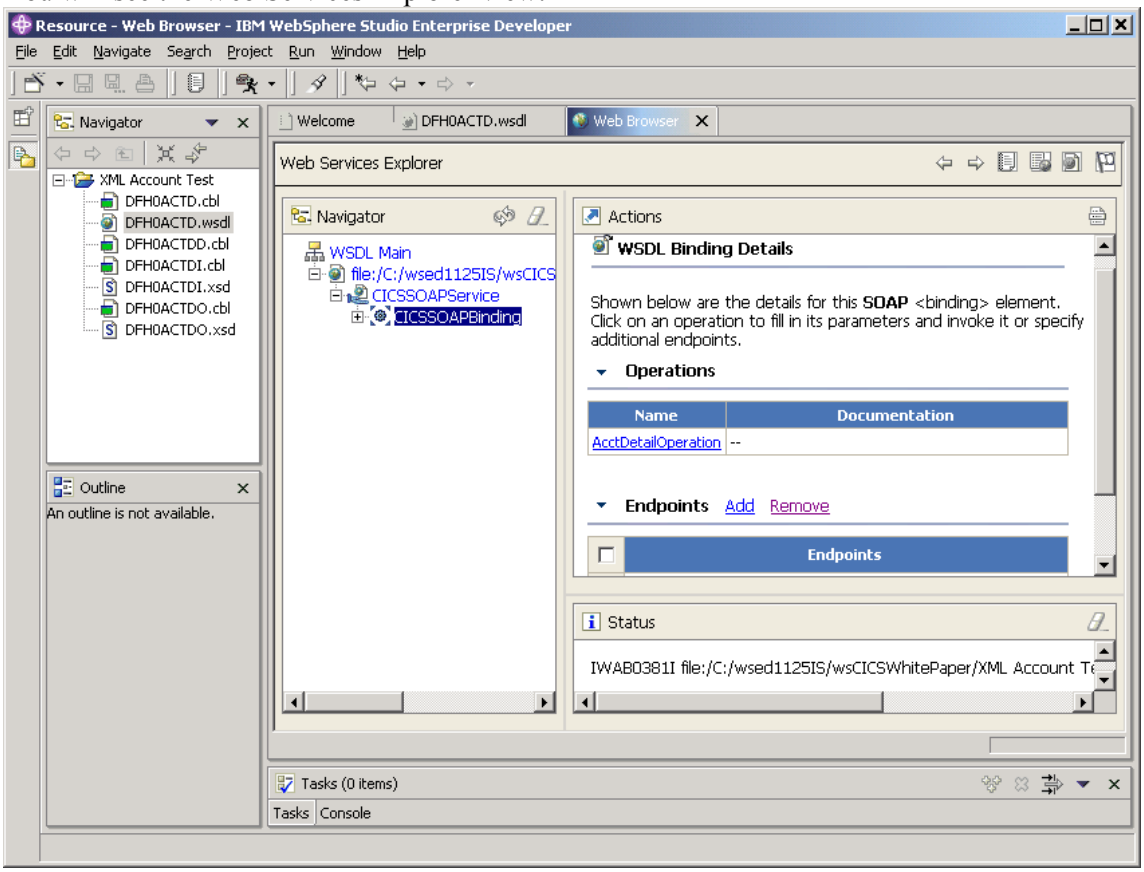


Figure 51, Web Services Explorer View

Navigate to the AcctDetailOperation in the Web Services Explorer Navigator view:

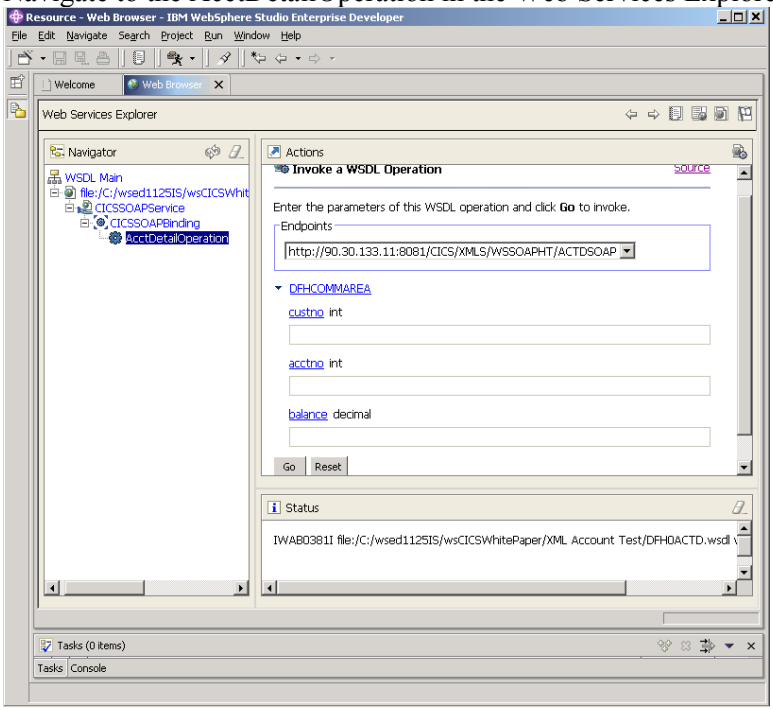


Figure 52, Web Services Explorer Navigator View

Enter customer number (for example, 4) and zeros in other fields (you need to do that if you specified the whole DFHCOMMAREA as the input parameter. Alternatively you could have unselected **acctno** and **balance** from the inbound messages structure selection page when you generated converters:

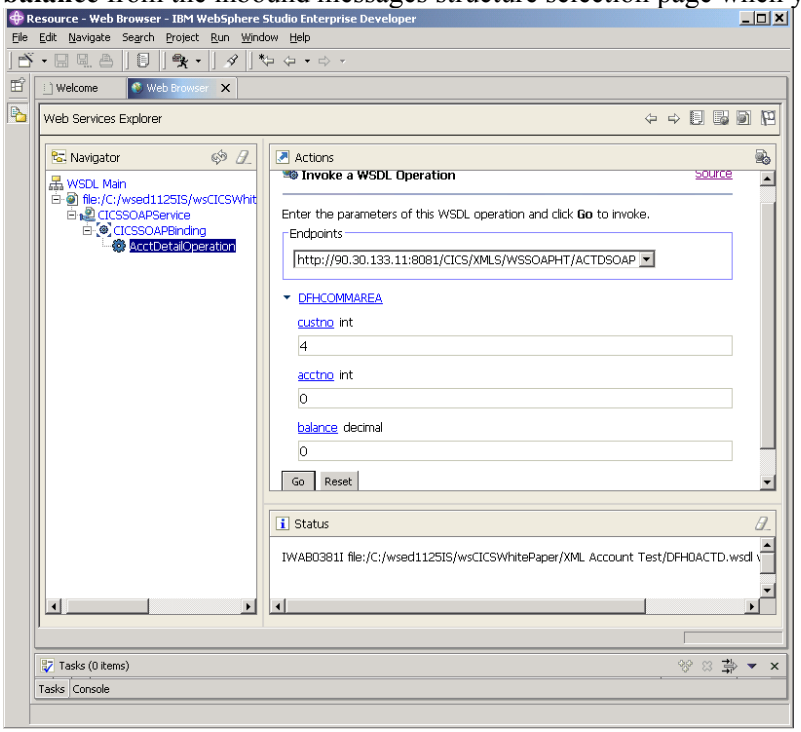


Figure 53, Entering the Customer Number

To test your web services, press the Go button in the invocation form.

The operation gets executed, and the results of executing your application as a Web Service are returned:

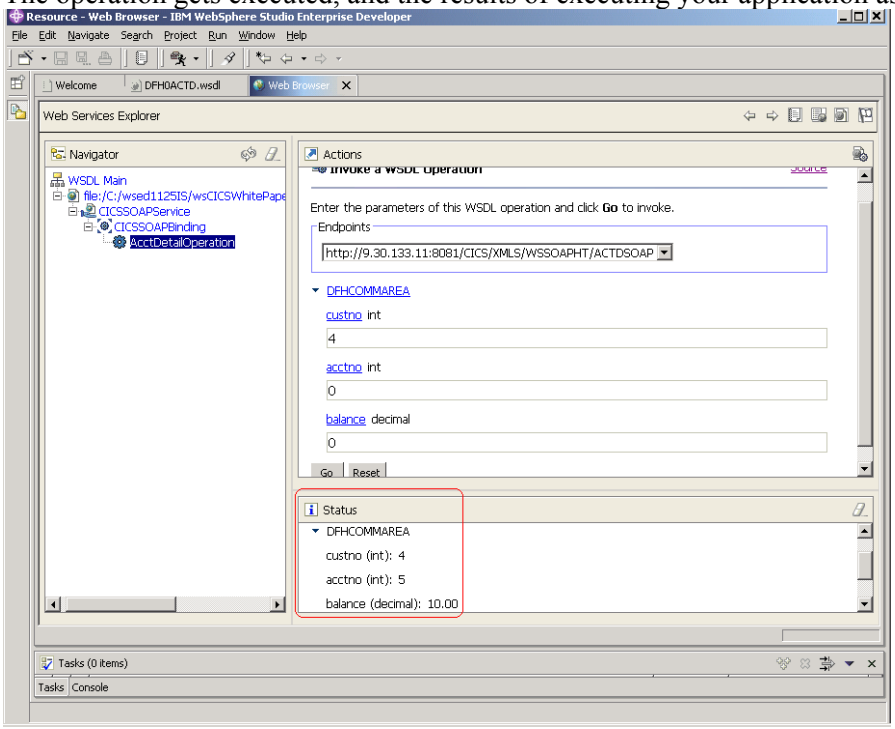


Figure 54, Results of Executing the Application

5.5 SOAP Error reporting

XML Enablement Converters, generated with the "CICS SOAP" driver type selected, return a SOAP Fault message to the service requester if an error is encountered during processing of a request. The message will always cite the senders request as the source of the problem and provide detail on the actual error that occurred.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<SOAP-ENV:Code>
<SOAP-ENV:Value>SOAP-ENV:Sender</SOAP-ENV:Value>
</SOAP-ENV:Code>
<SOAP-ENV:Reason>
<![CDATA[IGZ0282S XML to data structure conversion could not complete in program "ACTSOAPI"
because no element names in the XML document were recognized by the converter.]]>
</SOAP-ENV:Reason>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You can see the results of a SOAP fault in the Web Services Explorer by switching to the source view of the Status window:

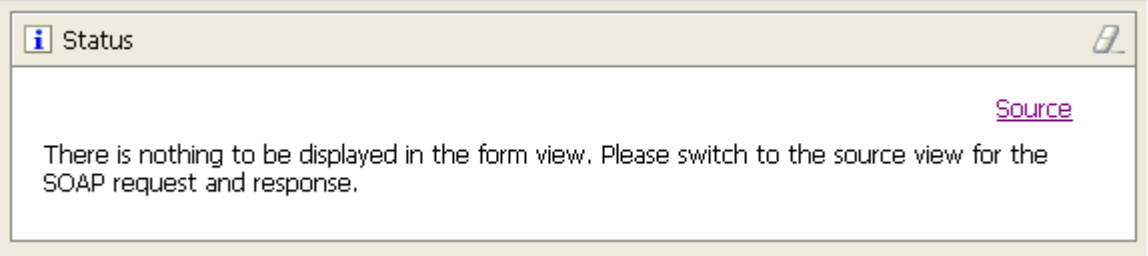


Figure 55, Results of a SOAP Fault

The fault message in the SOAP Response Envelope will look as follows:

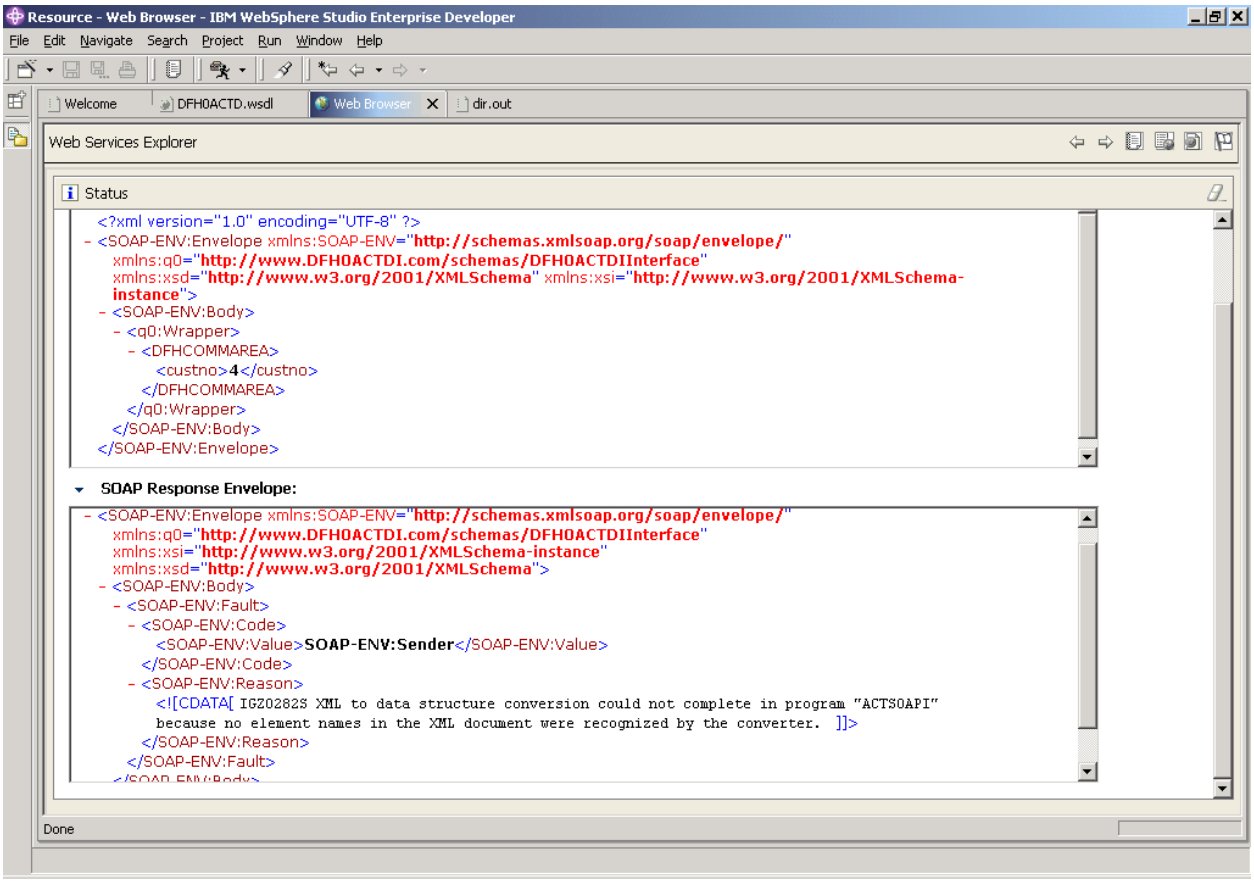


Figure 56, SOAP Response Envelope

6 Appendices

6.1 Appendix A. Pre-requisite software

The following software is required to develop and run sample applications described in this paper:

- CICS Transaction Server for OS/390 Version 2 Release 2 (program number 5697-E93) or later with the SOAP for CICS feature (HCAV100) installed.[1]
- IBM Database 2 Universal Database Server for OS/390 (DB2) Version 6 Release 1 (program number 5675-DB2) or later
- IBM Enterprise COBOL for z/OS and OS/390 Version 3 Release 1 (program number 5648-A25) or later with PTF for APAR PQ80513 (Available April 2003)
- IBM Language Environment® for OS/390 Version 2 Release 10 (program number 5647-A01) or later with PTF for APAR PQ65085 (Available Sept. 2002)
- IBM WebSphere Studio Enterprise Developer for Multiplatforms Version 5.1.2 (program number 5724-B67) or later
- OS/390 R8/R9/R10 and z/OS V1R1 support for Unicode ™ is required for the XML converters generated by WebSphere Studio Enterprise Developer, General Availability release[2].

[1] Soap for CICS (Feature HCAV100) is available free of charge at: http://www.ibm.com/cics/soap

[2] OS/390 R8/R9/R10 and z/OS V1R1 support for Unicode ™ can be obtained free of charge at https://www6.software.ibm.com/dl/os390/unicodespt-p . This support is integrated into z/OS Version 1 Release 2 and later.

6.2 Appendix B. Modifying COBOL Generator preferences

You access COBOL generator default preferences by selecting
Window > Preferences from the Workbench menu bar. The following dialog will appear.

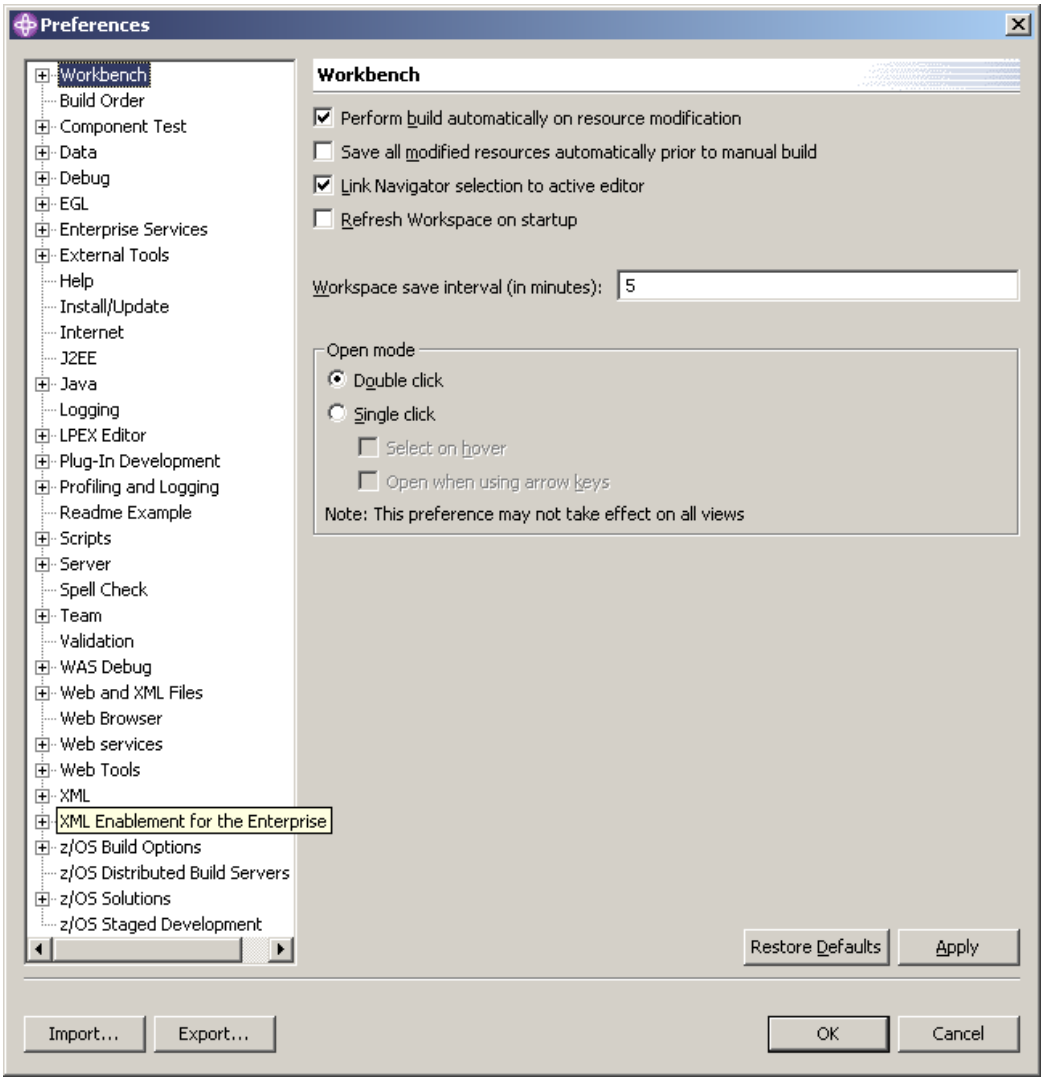


Figure 57, Workbench preferences dialog

Expand XML Enablement for the Enterprise, expand Generate XML Converter Wizard and then select the COBOL Generator page.

The following dialog will appear:

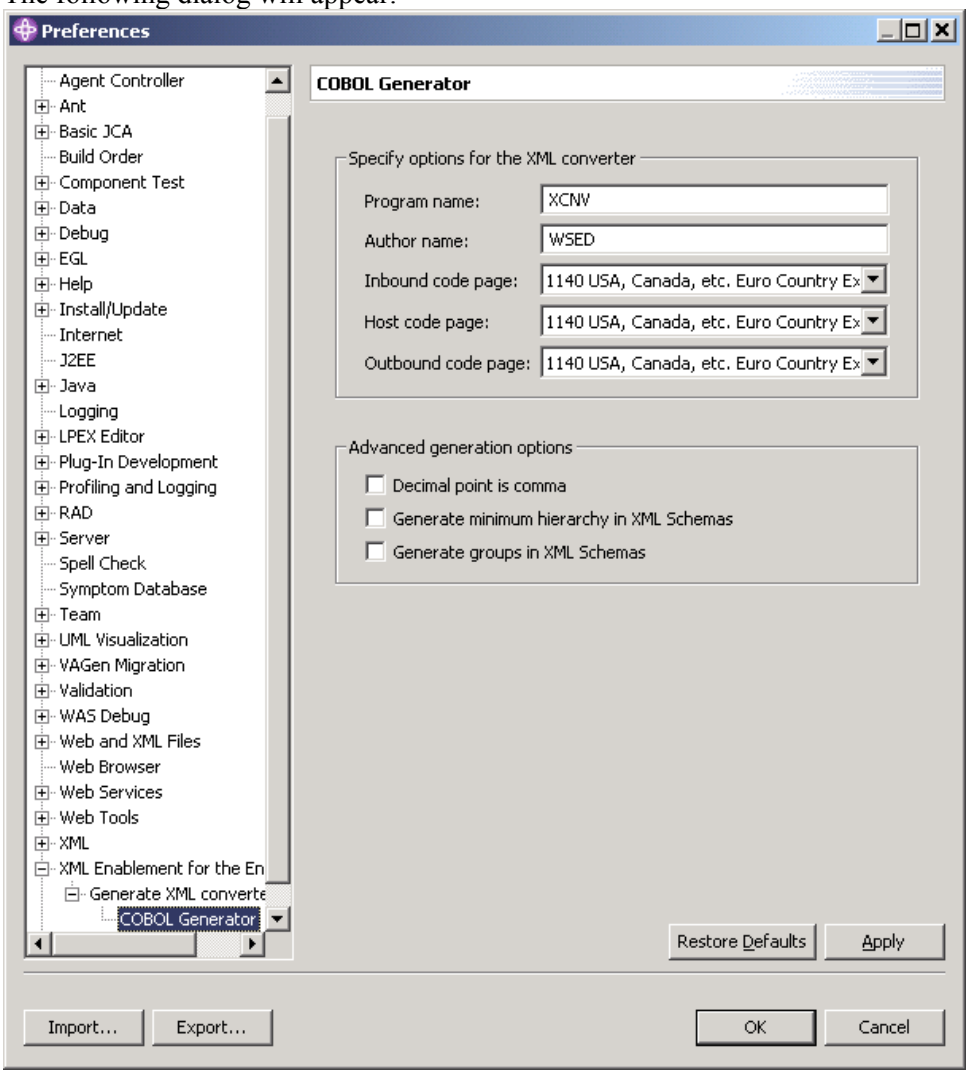


Figure 58, COBOL Generator preferences

Note that the entries on this page are very similar to the Options page of the COBOL converter generator wizard. In fact, this preferences page is where the defaults for the wizard come from. The only difference is the "Advanced generation options" section in which

- When "Decimal point is comma" is selected, the functions of the period and the comma are exchanged in PICTURE character strings and in numeric literals in a COBOL program.
- When "Generate minimum hierarchy in XML Schemas" is selected, the generated XML schemas will not contain the hierarchical structure that would reflect the original COBOL data structure nesting level unless it is required to disambiguate similarly named data items.
- When "Generate groups in XML Schemas" is specified, the generated XML Schemas may contain group model elements and references that are optional in the Web Services Interoperability Basic Profile 1.0 specification (see http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.htm). If you want your schemas to comply with WS-I BP 1.0 leave this option un-selected.

6.3 Appendix C. Modifying COBOL Importer preferences

You access COBOL importer default preferences by selecting
Window > Preferences from the Workbench menu bar. The following dialog will appear.

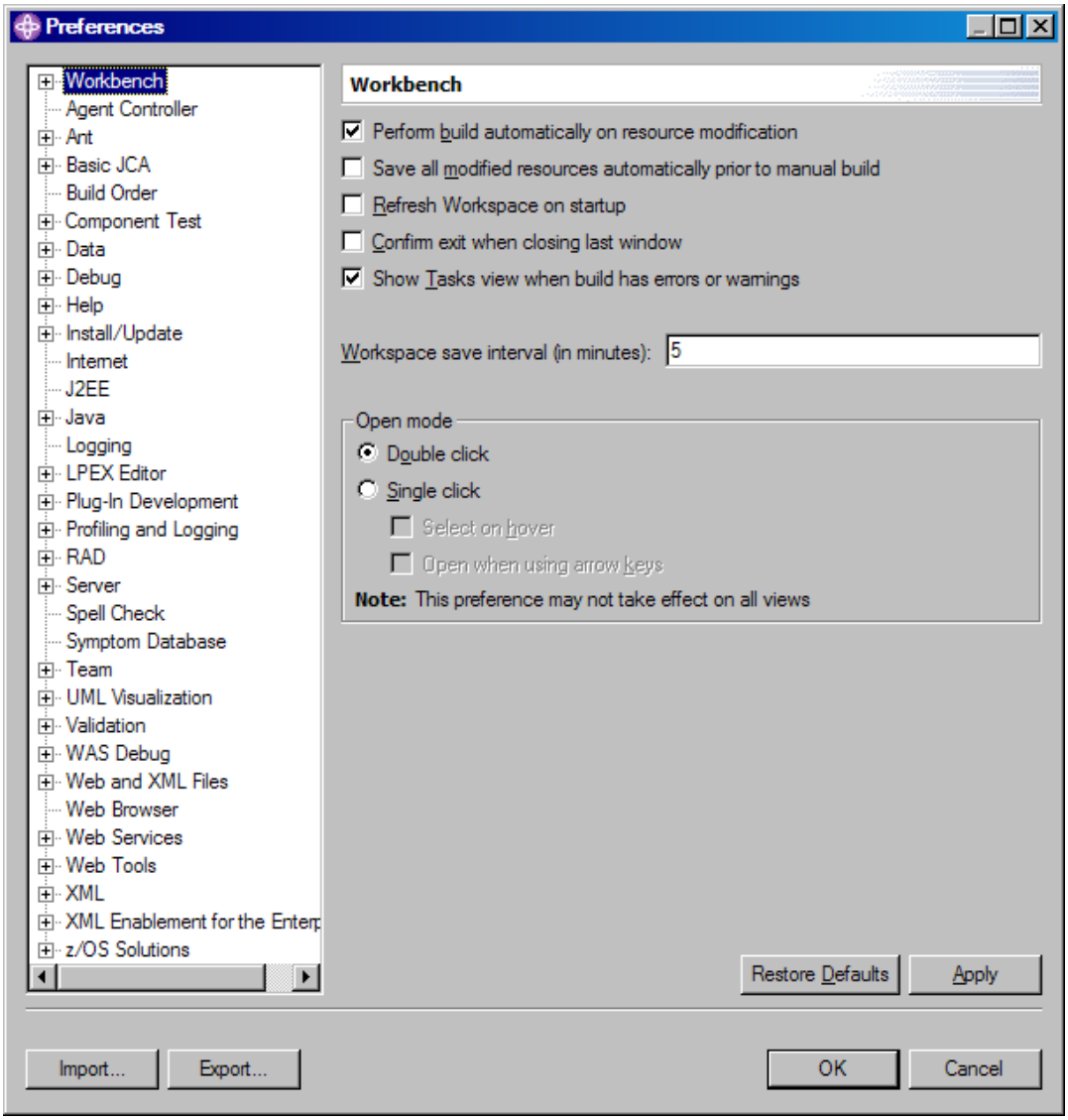


Figure 59, Workbench preferences dialog

Expand Basic JCA, expand importers and then select the COBOL page.

The following dialog will appear:

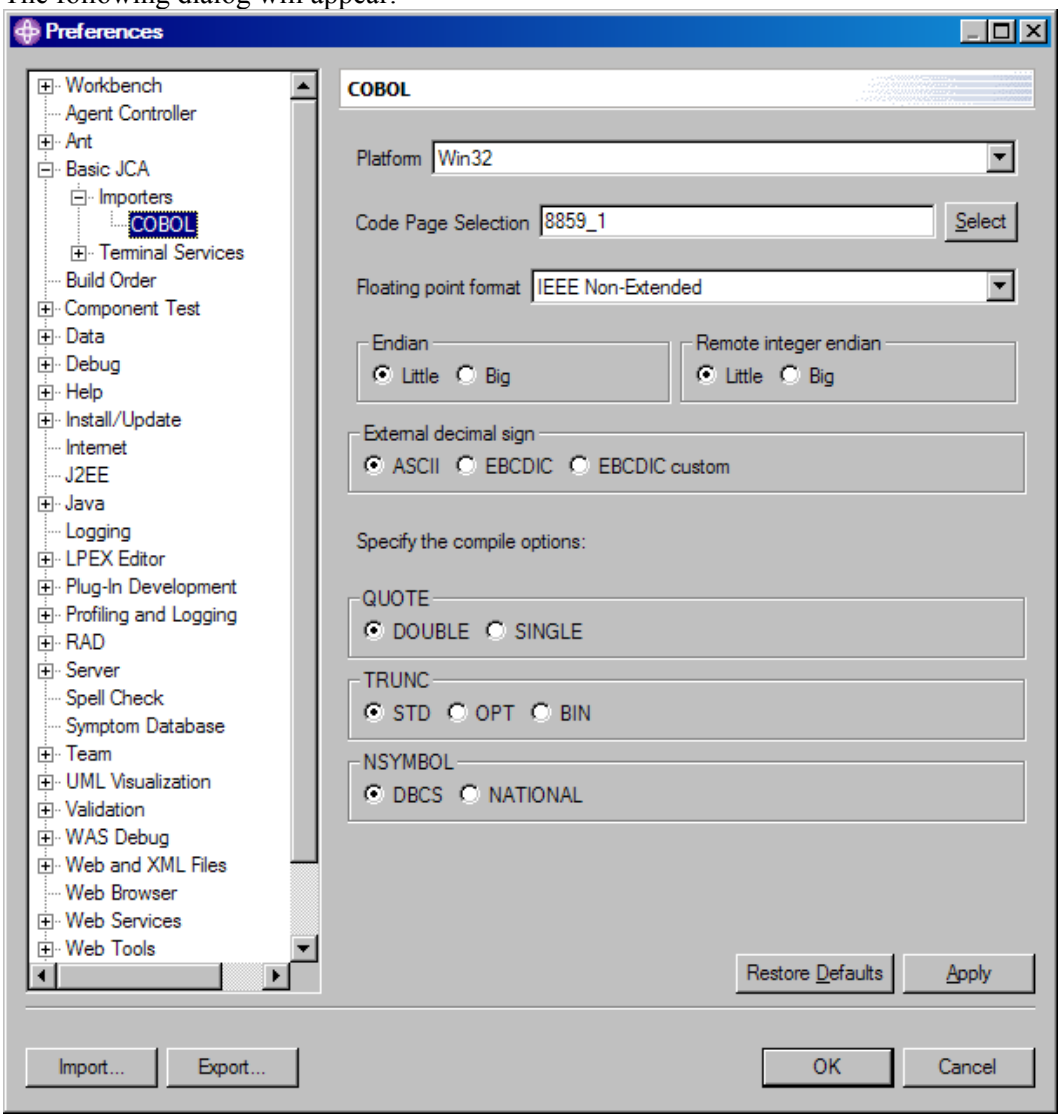


Figure 60, COBOL Importer preferences

This dialog allows you to change COBOL options that affect the way XML converters treat COBOL data items. Normally you should only modify the "Specify the compile options" section of this dialog, as the rest of the preferences do not apply to the XML converters.

6.4 Appendix D. XML Converter Interface

Both the inbound and outbound XML converters are invoked via a call statement. Arguments to the converters are a mixture of input and output parameters whose contents may be changed upon return from invocation. The call signature of the converters is displayed below:

```
CALL 'CONV' USING
              DATA-STRUCTURE   (input)
              XML-MESSAGE-LEN  (input + output)
              XML-MESSAGE-TEXT (input + output)
             (FEEDBACK-CODE or OMITTED) (output)
             RETURING
              CONVERTER-RETURN-CODE (output)
```

The above COBOL code is an example of a call to a converter. Input and output properties for each argument are displayed in parenthesized italics. Since the structure of each argument is unique, it must not vary from the description here.

DATA-STRUCTURE is a piece of storage whose structure is identical to that of the data structure that was nominated as the inbound data structure when the converter was generated. During an inbound conversion, DATA-STRUCTURE will be populated with values from the input XML document provided in the arguments XML-MESSAGE-LEN and XML-MESSAGE-TEXT. In the case of an outbound conversion, DATA-STRUCTURE is used to populate an XML message, whose properties are placed in the XML-MESSAGE-LEN and XML-MESSAGE-TEXT arguments.

FEEDBACK-CODE is a 12 byte Language Environment Condition Token that can be omitted by using the "OMITTED" keyword on the call. Choosing to omit this argument will cause any error encountered by the converter to be signaled as a severe condition containing information about the error. On the other hand not omitting this argument will cause the converter to simply place a condition token representing the error into FEEDBACK-CODE without signaling a condition. The structure of FEEDBACK-CODE is displayed below.

```
       1 FEEDBACK-CODE.
         2 CONDITION-TOKEN-VALUE.
           COPY CEEIGZCT.
          3 CASE-1-CONDITION-ID.
           4 SEVERITY    PIC S9(4) BINARY.
           4 MSG-NO      PIC S9(4) BINARY.
          3 CASE-2-CONDITION-ID
                REDEFINES CASE-1-CONDITION-ID.
           4 CLASS-CODE  PIC S9(4) BINARY.
           4 CAUSE-CODE  PIC S9(4) BINARY.
          3 CASE-SEV-CTL PIC X.
          3 FACILITY-ID  PIC XXX.
         2 I-S-INFO      PIC S9(9) BINARY.
```

More detailed information about the structure and use of this condition token can be found in the Language Environment Programming Guide.

CONVERTER-RETURN-CODE is an output only argument, which will contain one of two classes of return codes upon completion of the call. If the converter encounters an error within its own facilities, that is, not an error from the XML PARSE statement, then the Language Environment message ID associated with the error will be placed in the argument. The second class of return codes is the codes returned from the XML PARSE statement. These will occur in the case where something was syntactically incorrect in the input XML document. Note that this second class of errors only occurs during an inbound conversion.

6.5 Appendix E. Business applications program source

6.5.1 DFH0ACTD

```
      ***************************************************************  00600000
      *                                                             *  01300000
      * MODULE NAME = DFH0ACTD                                      *  02000000
      *                                                             *  02700000
      * DESCRIPTIVE NAME = %PRODUCT (Samples) EJB BankAccount Sample *  03400000
      *                                                             *  04100000
      *  %COPYRIGHT                                                 *  04800000
      **                                                            *  05500000
      * STATUS = %PLU0                                              *  06200000
      *                                                             *  06900000
      * TRANSACTION NAME = None                                     *  07600000
      *                                                             *  08300000
      * FUNCTION =                                                  *  09000000
      **     This module accesses the DB2 BankAccount tables for    *  09700000
      **     the customer details.                                  *  10400000
      **                                                            *  11100000
      *                                                             *  11800000
      * NOTES :                                                     *  12500000
      *  LANGUAGE  = COBOL 370                                      *  13200000
      *                                                             *  13900000
      *-------------------------------------------------------------*  14600000
      *                                                             *  15300000
      * ENTRY POINT = DFH0ACTD                                      *  16000000
      *-------------------------------------------------------------*  16700000
      *                                                             *  17400000
      * CHANGE ACTIVITY :                                           *  18100000
      **                                                            *  18800000
      *      $MOD(DFH0ACTD),COMP(SAMPLES),PROD(%PRODUCT):           *  19500000
      *                                                             *  20200000
      *   PN= REASON REL YYMMDD HDXXIII : REMARKS                   *  20900000
      *  $P0= D03094 %PL 011016 HDFVGMB : Rework the EJB BankAccount *  21600000
      ***************************************************************  22300000
       IDENTIFICATION DIVISION.                                        23000000
       PROGRAM-ID. DFH0ACTD.                                           23700000
       ENVIRONMENT DIVISION.                                           24400000
       DATA DIVISION.                                                  25100000
       WORKING-STORAGE SECTION.                                        25800000
                                                                       26500000
           EXEC SQL INCLUDE SQLCA  END-EXEC.                           27200000
                                                                       27900000
       01 TMP             PIC X(40) VALUE SPACES.                     28600000
       01  SQL-MESSAGE               VALUE SPACES.                    29300000
           05  MSG        PIC X(10).                                  30000000
           05  RC         PIC X(10).                                  30700000
       01  SQLERROR.                                                  31400000
           05  MSG2       PIC X(10) VALUE 'SQLERRM:  '.               32100000
           05  ERRM       PIC X(5).                                   32800000
           05  ERMC       PIC X(70) VALUE SPACES.                     33500000
       01  SQLSTAT.                                                   34200000
           05  MSG3       PIC X(10) VALUE 'SQLSTATE: '.               34900000
           05  SQLSTATT   PIC X(5).                                   35600000
       01  SQLERRORP.                                                 36300000
           05  MSG4       PIC X(10) VALUE 'SQLERRP:  '.               37000000
           05  SQLERP     PIC X(8).                                   37700000
                                                                       38400000
       01  ABEND-MESSAGE.                                             39100000
           05  MSG5       PIC X(12) VALUE 'ABEND CODE: '.             39800000
           05  ABEND-CODE PIC X(4).                                   40500000
                                                                       41200000
       01  HV-DATA.                                                   41900000
           05 HV-CUSTNO       PIC S9(9) COMP VALUE +0.                42600000
           05 HV-ACCTNO       PIC S9(9) COMP VALUE +0.                43300000
           05 HV-BALANCE      PIC S9(6)V9(2) COMP-3 VALUE +0.         44000000
                                                                       44700000
       LINKAGE SECTION.                                                45400000
       01  DFHCOMMAREA.                                               46100000
           05  CUSTNO     PIC S99999.                                 46800000
           05  ACCTNO     PIC S99999.                                 47500000
           05  BALANCE    PIC S9999V99.                               48200000
                                                                       48900000
```

```
       PROCEDURE DIVISION.                                                      49600000
       START-PARA.                                                              50300000
                                                                                51000000
           MOVE 999999999 TO ACCTNO                                             51700000
           MOVE 'SQLCODE: ' TO MSG.                                             52400000
           MOVE 'DFH0ACTD PROGRAM STARTED. ' TO TMP.                            53100000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                                    53800000
             FROM(TMP)                                                          54500000
             LENGTH(40)                                                         55200000
             END-EXEC.                                                          55900000
                                                                                56600000
           MOVE CUSTNO TO HV-CUSTNO.                                            57300000
           MOVE 'SEARCHING WITH CUST NO:' TO TMP.                               58000000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                                    58700000
             FROM(TMP)                                                          59400000
             LENGTH(40)                                                         60100000
             END-EXEC.                                                          60800000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                                    61500000
             FROM(CUSTNO)                                                       62200000
             LENGTH(5)                                                          62900000
             END-EXEC.                                                          63600000
           EXEC CICS HANDLE ABEND                                               64300000
             LABEL(ABEND-PARA)                                                  65000000
             END-EXEC.                                                          65700000
                                                                                66400000
           EXEC SQL SELECT ACCT_NUMBER, BALANCE                                 67100000
                    INTO :HV-ACCTNO, :HV-BALANCE                                67800000
                    FROM ACCOUNT                                                68500000
                    WHERE BALANCE IN                                            69200000
                    (SELECT MAX(BALANCE) FROM ACCOUNT                           69900000
                    WHERE CUST_ID = :HV-CUSTNO) END-EXEC.                       70600000
                                                                                71300000
           MOVE SQLCODE TO RC.                                                  72000000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                                    72700000
             FROM(SQL-MESSAGE)                                                  73400000
             LENGTH(20)                                                         74100000
             END-EXEC.                                                          74800000
                                                                                75500000
           MOVE SQLERRML TO ERRM.                                               76200000
           MOVE SQLERRMC TO ERMC.                                               76900000
           EXEC CICS WRITEQ TD QUEUE ('CSMT') FROM (SQLERROR)                   77600000
             LENGTH(85) END-EXEC.                                               78300000
           MOVE SQLERRP TO SQLERP.                                              79000000
           EXEC CICS WRITEQ TD QUEUE ('CSMT') FROM (SQLERRORP)                  79700000
             LENGTH(18) END-EXEC.                                               80400000
           MOVE SQLSTATE TO SQLSTATT.                                           81100000
           EXEC CICS WRITEQ TD QUEUE ('CSMT') FROM (SQLSTAT)                    81800000
             LENGTH(15) END-EXEC.                                               82500000
                                                                                83200000
           IF  SQLCODE EQUAL ZERO                                               83900000
               MOVE HV-ACCTNO TO ACCTNO                                         84600000
               MOVE HV-BALANCE TO BALANCE                                       85300000
           END-IF.                                                              86000000
           GO TO RETURN-PARA.                                                   86700000
       ABEND-PARA.                                                              87400000
           EXEC CICS HANDLE ABEND                                               88100000
             CANCEL                                                             88800000
             END-EXEC.                                                          89500000
           EXEC CICS ASSIGN                                                     90200000
             ABCODE(ABEND-CODE)                                                 90900000
             END-EXEC.                                                          91600000
           EXEC CICS WRITEQ TD QUEUE ('CSMT') FROM (ABEND-MESSAGE)              92300000
             LENGTH(16) END-EXEC.                                               93000000
       RETURN-PARA.                                                             93700000
           MOVE 'DFH0ACTD PROGRAM STOPPED.' TO TMP.                             94400000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                                    95100000
             FROM(TMP)                                                          95800000
             LENGTH(40)                                                         96500000
             END-EXEC.                                                          97200000
                                                                                97900000
           EXEC CICS RETURN                                                     98600000
           END-EXEC.                                                            99300000
```

6.5.2 DFH0CSTD

```
      ***************************************************************  00600000
      *                                                             *  01200000
      * MODULE NAME = DFH0CSTD                                      *  01800000
      *                                                             *  02400000
      * DESCRIPTIVE NAME = %PRODUCT (Samples) EJB BankAccount Sample *  03000000
      *                                                             *  03600000
      *  %COPYRIGHT                                                 *  04200000
      **                                                            *  04800000
      * STATUS = %PLU0                                              *  05400000
      *                                                             *  06000000
      * TRANSACTION NAME = None                                     *  06600000
      *                                                             *  07200000
      * FUNCTION =                                                  *  07800000
      **     This module accesses the DB2 BankAccount tables for     *  08400000
      **     the account details.                                    *  09000000
      **                                                            *  09600000
      *                                                             *  10200000
      * NOTES :                                                     *  10800000
      *  LANGUAGE  = COBOL 370                                      *  11400000
      *                                                             *  12000000
      *-------------------------------------------------------------*  12600000
      *                                                             *  13200000
      * ENTRY POINT = DFH0CSTD                                      *  13800000
      *-------------------------------------------------------------*  14400000
      *                                                             *  15000000
      * CHANGE ACTIVITY :                                           *  15600000
      **                                                            *  16200000
      *      $MOD(DFH0CSTD),COMP(SAMPLES),PROD(%PRODUCT):           *  16800000
      *                                                             *  17400000
      *   PN= REASON REL YYMMDD HDXXIII : REMARKS                   *  18000000
      *  $P0= D03094 %PL 011016 HDFVGMB : Rework the EJB BankAccount *  18600000
      ***************************************************************  19200000
       IDENTIFICATION DIVISION.                                        19800000
       PROGRAM-ID. DFH0CSTD.                                           20400000
       ENVIRONMENT DIVISION.                                           21000000
       DATA DIVISION.                                                  21600000
       WORKING-STORAGE SECTION.                                        22200000
                                                                       22800000
                                                                       23400000
           EXEC SQL INCLUDE SQLCA  END-EXEC.                           24000000
                                                                       24600000
       01 TMP             PIC X(40) VALUE SPACES.                     25200000
       01  SQL-MESSAGE               VALUE SPACES.                    25800000
           05  MSG        PIC X(10).                                  26400000
           05  RC         PIC X(10).                                  27000000
       01  SQLERROR.                                                  27600000
           05 MSG2        PIC X(10) VALUE 'SQLERRM:  '.               28200000
           05 ERRM        PIC X(5).                                   28800000
           05 ERMC        PIC X(70) VALUE SPACES.                     29400000
       01  SQLSTAT.                                                   30000000
           05 MSG3        PIC X(10) VALUE 'SQLSTATE: '.               30600000
           05 SQLSTATT    PIC X(5).                                   31200000
       01  SQLERRORP.                                                 31800000
           05 MSG4        PIC X(10) VALUE 'SQLERRP:  '.               32400000
           05 SQLERP      PIC X(8).                                   33000000
       01  ABEND-MESSAGE.                                             33600000
           05  MSG5       PIC X(12) VALUE 'ABEND CODE: '.             34200000
           05  ABEND-CODE PIC X(4)  VALUE SPACES.                     34800000
                                                                       35400000
       01  HV-DATA.                                                   36000000
           05 HV-CUSTNO       PIC S9(9) COMP.                         36600000
           05 HV-FIRSTNAME    PIC X(15).                              37200000
           05 HV-LASTNAME     PIC X(25).                              37800000
           05 HV-ADDRESS      PIC X(20).                              38400000
           05 HV-CITY         PIC X(20).                              39000000
           05 HV-STATE        PIC X(10).                              39600000
           05 HV-COUNTRY      PIC X(15).                              40200000
                                                                       40800000
                                                                       41400000
       LINKAGE SECTION.                                                42000000
       01  DFHCOMMAREA.                                               42600000
           05  CUSTNO     PIC S99999.                                 43200000
           05  LASTNAME   PIC A(25).                                  43800000
           05  FIRSTNAME  PIC A(15).                                  44400000
           05  ADDRESS1   PIC X(20).                                  45000000
           05  CITY       PIC A(20).                                  45600000
           05  STATE      PIC A(10).                                  46200000
           05  COUNTRY    PIC X(15).                                  46800000
                                                                       47400000
       PROCEDURE DIVISION.                                             48000000
       START-PARA.                                                     48600000
                                                                       49200000
```

```
           INITIALIZE HV-DATA.                                          49800000
           INITIALIZE SQLCA.                                            50400000
                                                                        51000000
           MOVE 'SQLCODE: ' TO MSG.                                     51600000
           MOVE 'DFH0CSTD PROGRAM STARTED.' TO TMP.                     52200000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                            52800000
             FROM(TMP)                                                  53400000
             LENGTH(40)                                                 54000000
             END-EXEC.                                                  54600000
                                                                        55200000
           MOVE CUSTNO TO HV-CUSTNO.                                    55800000
           MOVE 'SEARCHING WITH CUST #:' TO TMP.                        56400000
                                                                        57000000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                            57600000
             FROM(TMP)                                                  58200000
             LENGTH(40)                                                 58800000
             END-EXEC.                                                  59400000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                            60000000
             FROM(CUSTNO)                                               60600000
             LENGTH(5)                                                  61200000
             END-EXEC.                                                  61800000
                                                                        62400000
           EXEC CICS HANDLE ABEND                                       63000000
             LABEL(ABEND-PARA)                                          63600000
             END-EXEC.                                                  64200000
                                                                        64800000
           EXEC SQL SELECT CUST_LN, CUST_FN, CUST_ADDR, CUST_CITY,      65400000
                    CUST_ST, CUST_CTRY INTO :HV-LASTNAME, :HV-FIRSTNAME,66000000
                    :HV-ADDRESS, :HV-CITY, :HV-STATE, :HV-COUNTRY       66600000
                    FROM CUSTOMER WHERE CUST_ID = :HV-CUSTNO            67200000
                    END-EXEC.                                           67800000
                                                                        68400000
           MOVE SQLCODE TO RC.                                          69000000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                            69600000
             FROM(SQL-MESSAGE)                                          70200000
             LENGTH(20)                                                 70800000
             END-EXEC.                                                  71400000
                                                                        72000000
           MOVE SQLERRML TO ERRM.                                       72600000
           MOVE SQLERRMC TO ERMC.                                       73200000
           EXEC CICS WRITEQ TD QUEUE('CSMT') FROM(SQLERROR) LENGTH(85)  73800000
             END-EXEC.                                                  74400000
           MOVE SQLERRP TO SQLERP.                                      75000000
           EXEC CICS WRITEQ TD QUEUE('CSMT') FROM(SQLERRORP) LENGTH(18) 75600000
             END-EXEC.                                                  76200000
           MOVE SQLSTATE TO SQLSTATT.                                   76800000
           EXEC CICS WRITEQ TD QUEUE('CSMT') FROM(SQLSTAT) LENGTH(15)   77400000
           END-EXEC.                                                    78000000
           IF  SQLCODE EQUAL ZERO                                       78600000
               MOVE HV-FIRSTNAME TO FIRSTNAME                           79200000
               MOVE HV-LASTNAME TO LASTNAME                             79800000
               MOVE HV-ADDRESS TO ADDRESS1                              80400000
               MOVE HV-CITY TO CITY                                     81000000
               MOVE HV-STATE TO STATE                                   81600000
               MOVE HV-COUNTRY TO COUNTRY                               82200000
           ELSE                                                         82800000
               MOVE 'DB2 ERROR - SEE CICS LOG' TO LASTNAME              83400000
               MOVE SQL-MESSAGE  TO ADDRESS1                            84000000
           END-IF.                                                      84600000
           GO TO RETURN-PARA.                                           85300000
       ABEND-PARA.                                                      86000000
           EXEC CICS HANDLE ABEND                                       86700000
             CANCEL                                                     87400000
             END-EXEC.                                                  88100000
           EXEC CICS ASSIGN                                             88800000
             ABCODE(ABEND-CODE)                                         89500000
             END-EXEC.                                                  90200000
           EXEC CICS WRITEQ TD QUEUE ('CSMT') FROM (ABEND-MESSAGE)      90900000
             LENGTH(16) END-EXEC.                                       91600000
            MOVE 'CICS TRANSACTION ABENDED ' TO LASTNAME.               92300000
            MOVE ABEND-MESSAGE  TO ADDRESS1.                            93000000
       RETURN-PARA.                                                     93700000
           MOVE 'DFH0CSTD PROGRAM STOPPED.' TO TMP.                     94400000
           EXEC CICS WRITEQ TD QUEUE('CSMT')                            95100000
             FROM(TMP)                                                  95800000
             LENGTH(40)                                                 96500000
             END-EXEC.                                                  97200000
                                                                        97900000
           EXEC CICS RETURN                                             98600000
           END-EXEC.                                                    99300000
```

7 Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
P.O. Box 12195, Dept. TL3B/B503/B313
3039 Cornwallis Rd.
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.
(C) Copyright IBM Corp. _enter the year or years_. All rights reserved.

7.1 Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

7.2 Trademarks and service marks

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

- CICS(R)
- DB2(R)
- IBM(R)
- OS/390(R)
- WebSphere(R)

[1]Java(TM) and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.