

CICS Web Services as a Provider and Requestor

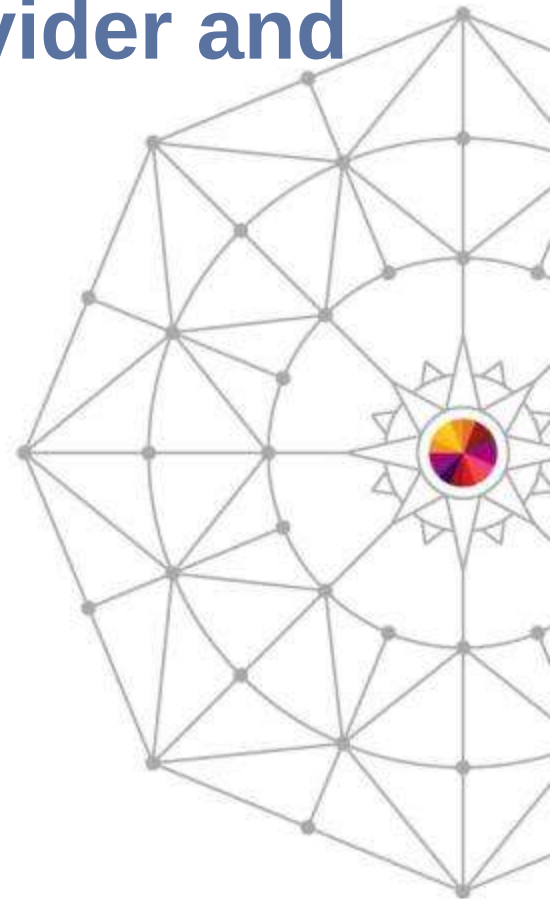
Ezriel Gross

Circle Software Incorporated

March 13th, 2013 (Thu)

3:00pm – 4:00pm

Session 14827



Agenda

- Introduction to web services in general, and in CICS
- Four methods for creating a web service provider in CICS:
 1. CICS web services assistant
 2. Rational Developer for System z (RDz) with interpretive runtime XML conversion
 3. RDz, with compiled runtime XML conversion
 4. RDz Service Flow Modeler (SFM)
- Two methods for creating a web service requester in CICS:
 1. CICS web services assistant
 2. RDz
- Diagnosing web services in CICS

Terms

Web service

- A software system designed to support interoperable machine-to-machine interaction over a network
- It has an interface described in a machine-processable format (specifically **WSDL**)
- Other systems interact with *[it ...]* using **SOAP** messages, typically conveyed using **HTTP** *[...]*

or MQ, JCA... in the examples presented here, we will use HTTP

Complete your session evaluations online at www.SHARE.org/AnaheimEval

WSDL

- *[Web Service Description Language is an XML vocabulary that] describes [...] the messages that are exchanged between the requester and provider*

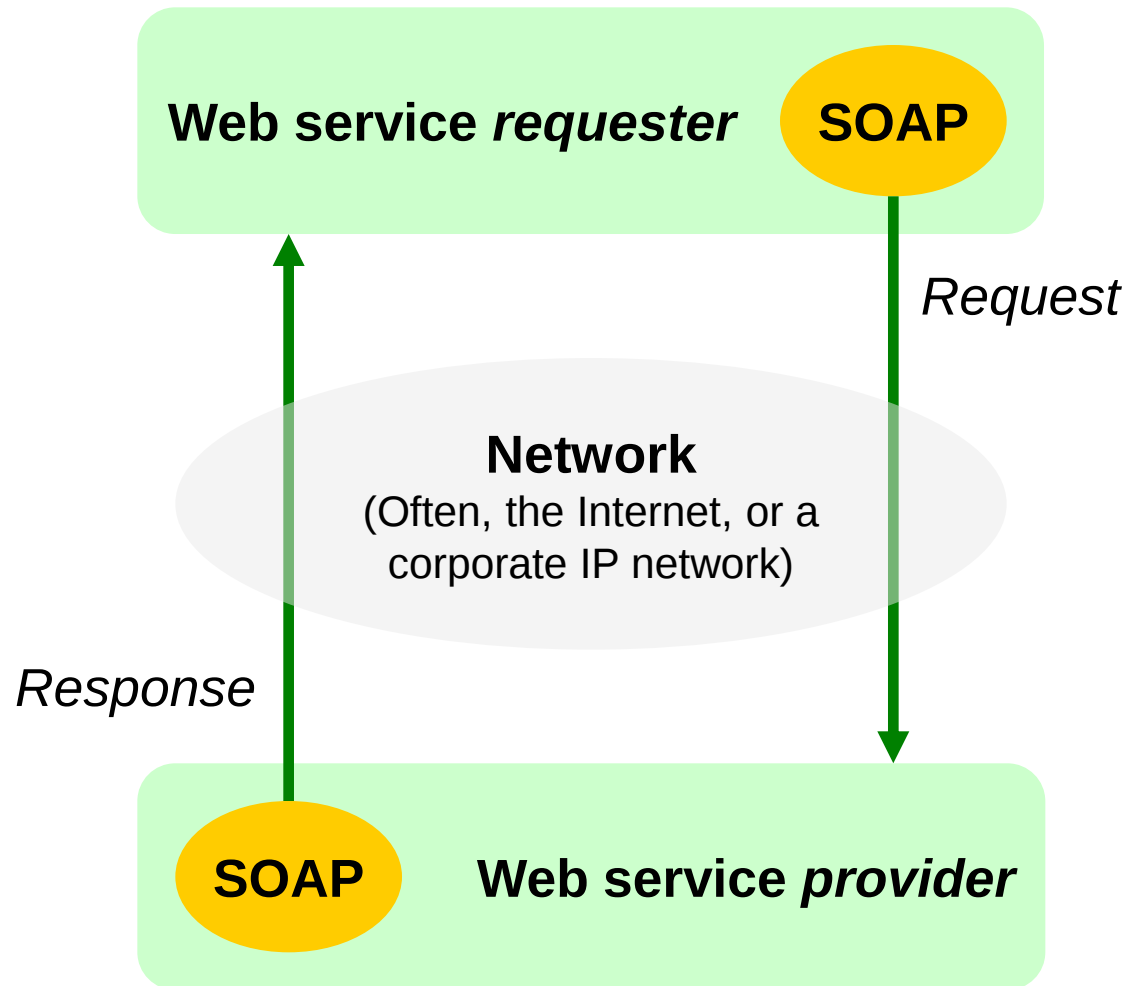
SOAP

- *[A ...] framework for packaging and exchanging XML messages*

Source: *Web Services Architecture*

<http://www.w3.org/TR/ws-arch/>

Basic concept



Example SOAP request

XML defined by the SOAP standard

```
<soapenv:Envelope
  xmlns="http://www.PAYBUS.PAYCOM1.Request.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <PAYBUSOperation>
      <ws_payroll_data>
        <ws_request>DISP</ws_request>
        <ws_key>
          <ws_department>1</ws_department>
          <ws_employee_no>00001</ws_employee_no>
        </ws_key>
      </ws_payroll_data>
      ...some markup omitted for brevity...
    </PAYBUS1Operation>
  </soapenv:Body>
</soapenv:Envelope>
```

Web service-specific XML
(contents of the SOAP Body) is
described in a WSDL file

In plain English:

Please “display” payroll data for
employee number 1
in department 1

Example SOAP response

```
<soapenv:Envelope
  xmlns="http://www.PAYBUS.PAYCOM1.Request.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <PAYBUSOperationResponse>
      <ws_payroll_data>
        <ws_request>DISP</ws_request>
        <ws_key>
          <ws_department>1</ws_department>
          <ws_employee_no>00001</ws_employee_no>
        </ws_key>
        <ws_name>CIRCLE COMPUTER 1 </ws_name>
        <ws_addr1>65 WILLOWBROOK BLVD </ws_addr1>
        <ws_addr2>4TH FLOOR</ws_addr2>
        <ws_addr3>WAYNE, NJ 07470 </ws_addr3>
        <ws_phone_no>890-9331</ws_phone_no>
        <ws_timestamp/>
        <ws_salary>50000.00</ws_salary>
        <ws_start_date>12312008</ws_start_date>
        <ws_remarks>CIRCLE IS MAGIC </ws_remarks>
        ...some markup omitted for brevity...
      </PAYBUSOperationResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Response details

Web Service Description Language (WSDL) file

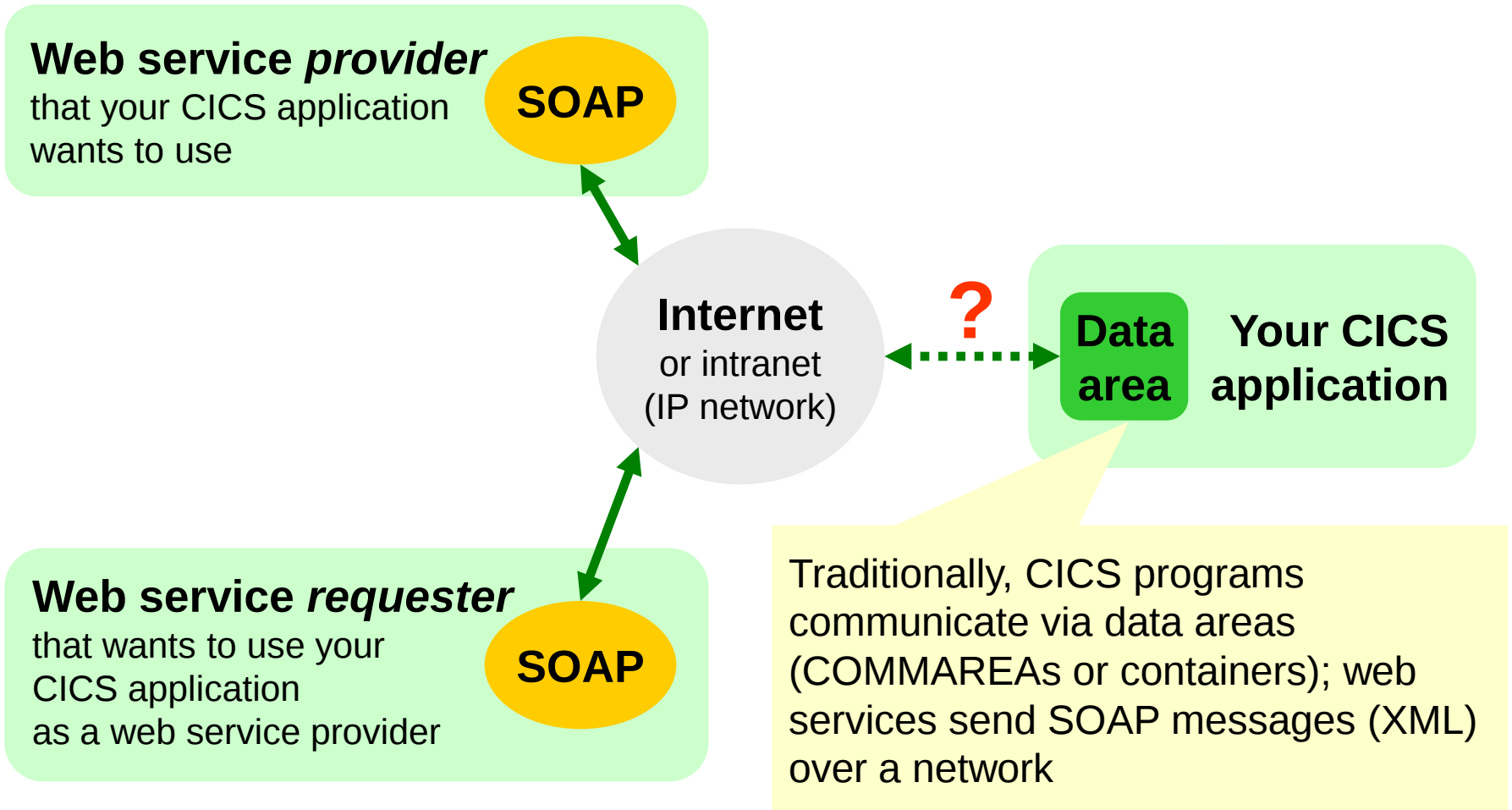
- WSDL 1.1 (see below) or 2.0: generated by CICS web services assistant or RDz (if you don't have one)
- Describes the request/response message XML (schema); groups messages into operations on an abstract port; binds the operations to a message transport; specifies the web service address

```
<definitions ... >
  <types>
    <xsd:schema ... > ... </xsd:schema>
    <xsd:schema ... > ... </xsd:schema>
  </types>
  <message name="PAYBUSOperationResponse">
    <part element="resns:PAYBUSOperationResponse" name="ResponsePart"/>
  </message>
  <message name="PAYBUSOperationRequest">
    <part element="reqns:PAYBUSOperation" name="RequestPart"/>
  </message>
```

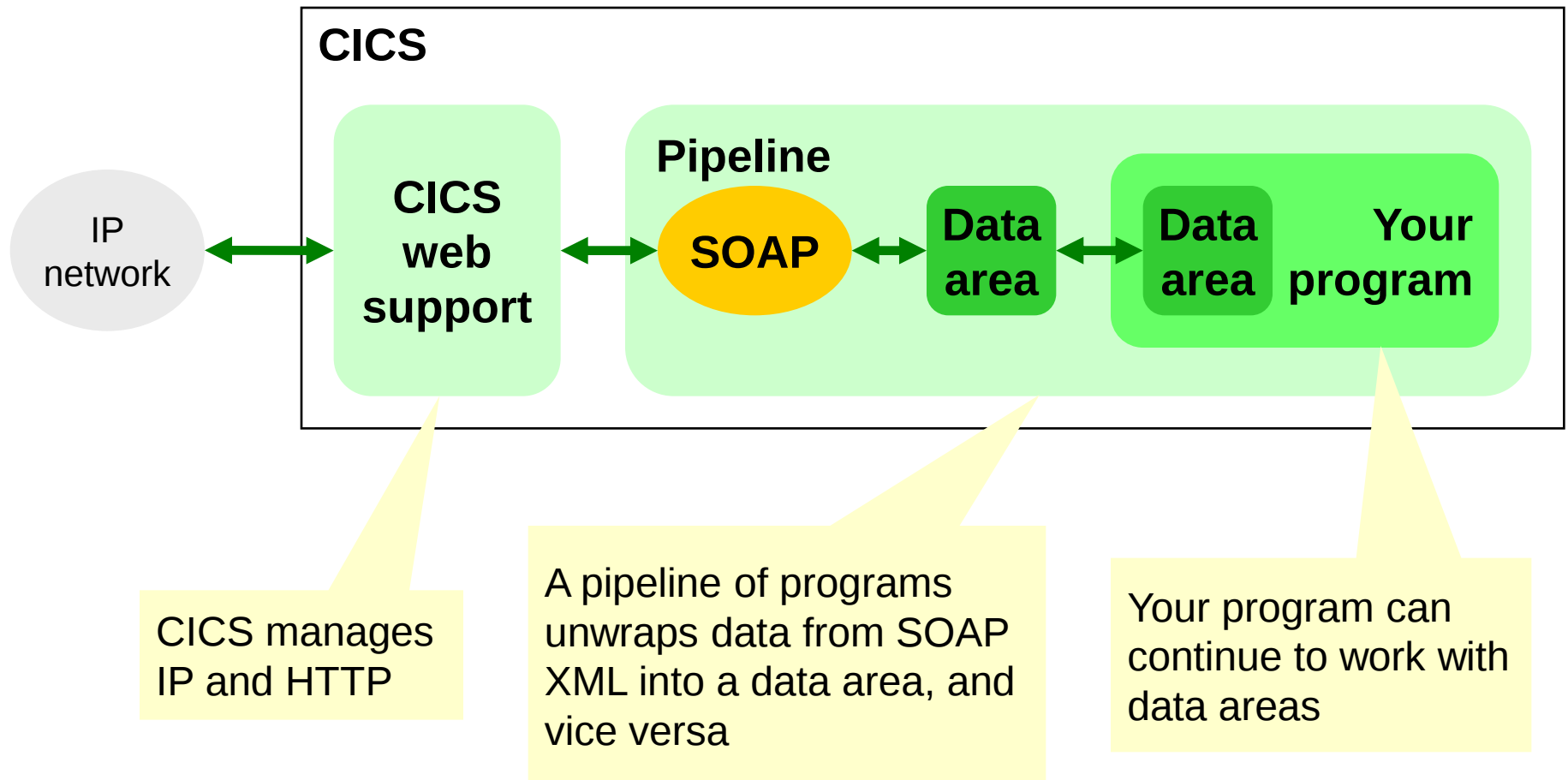
WSDL 1.1 file, continued

```
<portType name="PAYBUSPort">
  <operation name="PAYBUSOperation">
    <input message="tns:PAYBUSOperationRequest" name="PAYBUSOperationRequest"/>
    <output message="tns:PAYBUSOperationResponse" name="PAYBUSOperationResponse"/>
  </operation>
</portType>
<binding name="PAYBUSHTTPSoapBinding" type="tns:PAYBUSPort">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="PAYBUSOperation">
    <soap:operation soapAction="" style="document"/>
    <input name="PAYBUSOperationRequest">
      <soap:body parts="RequestPart" use="literal"/>
    </input>
    <output name="PAYBUSOperationResponse">
      <soap:body parts="ResponsePart" use="literal"/>
    </output>
  </operation>
</binding>
<service name="PAYBUSService">
  <port binding="tns:PAYBUSHTTPSoapBinding" name="PAYBUSPort">
    <soap:address location="http://my-server:my-port/paybus1"/>
  </port>
</service>
</definitions>
```

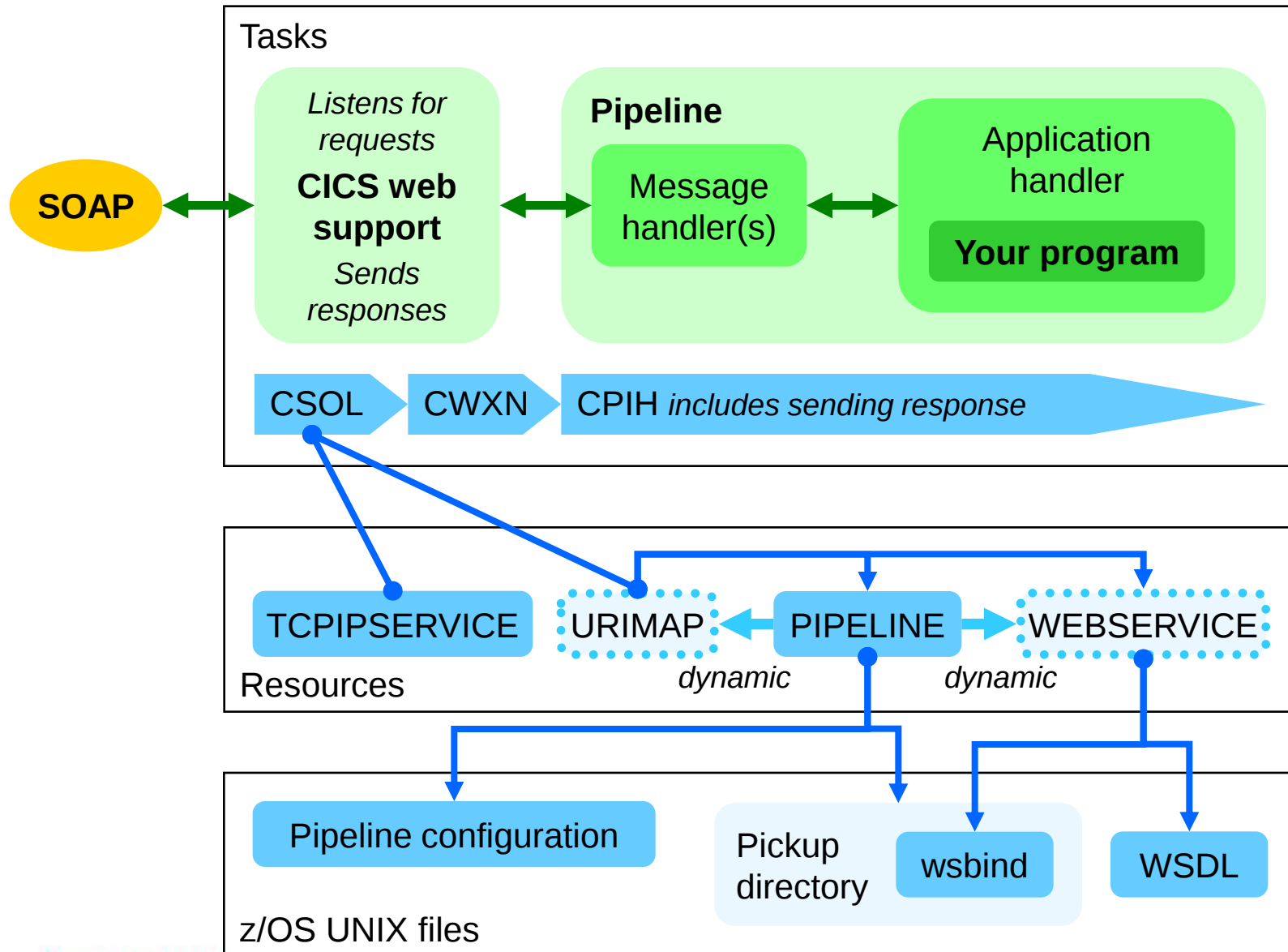
Problem



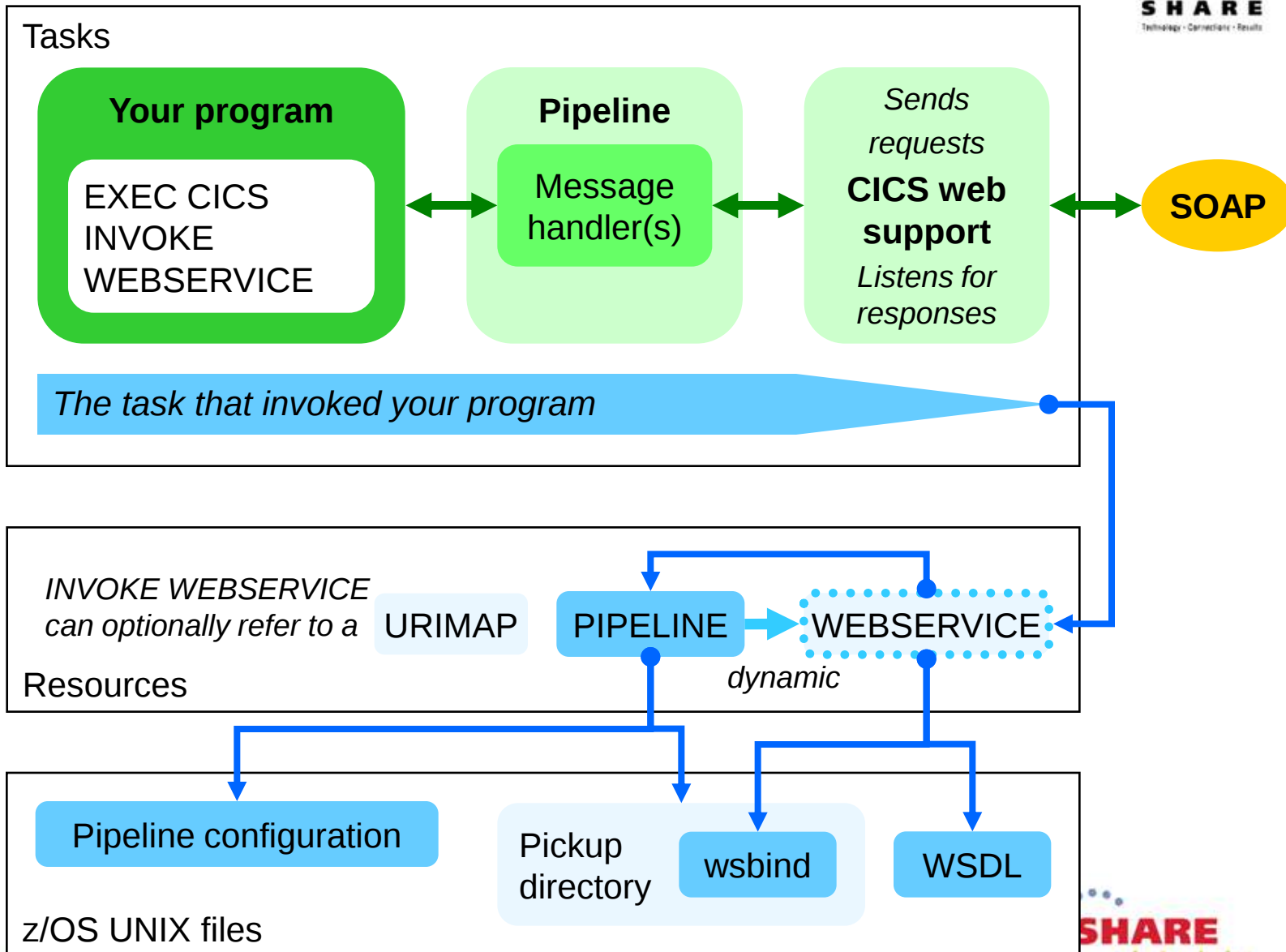
Solution



CICS as a web service provider



CICS as a web service requester



CICS resources

- You must manually create:
 - **Provider only:**
TCPIP SERVICE: Specifies which port to listen to for requests. (This assumes HTTP message transport. For WebSphere MQ, you would create an MQCONN.)
 - **PIPELINE:** Points to a pipeline configuration file, which specifies the sequence of handler programs in the pipeline.
- CICS dynamically creates when PIPELINE is installed (or when you run the PIPELINE SCAN command):
 - **Provider only:**
URIMAP: Specifies which pipeline and web service to use for this request. (For a requester, the INVOKE (WEB)SERVICE can optionally refer to a URIMAP for the provider address.)
 - **WEBSERVICE:** Points to a WSDL file and a wsbind file.

Pipeline configuration file

- Defines the handlers that constitute the pipeline (in these examples, the single handler wraps/unwraps the contents of the SOAP message body in the SOAP envelope)
- If you do not require special processing, you can use these IBM-supplied sample files unchanged:

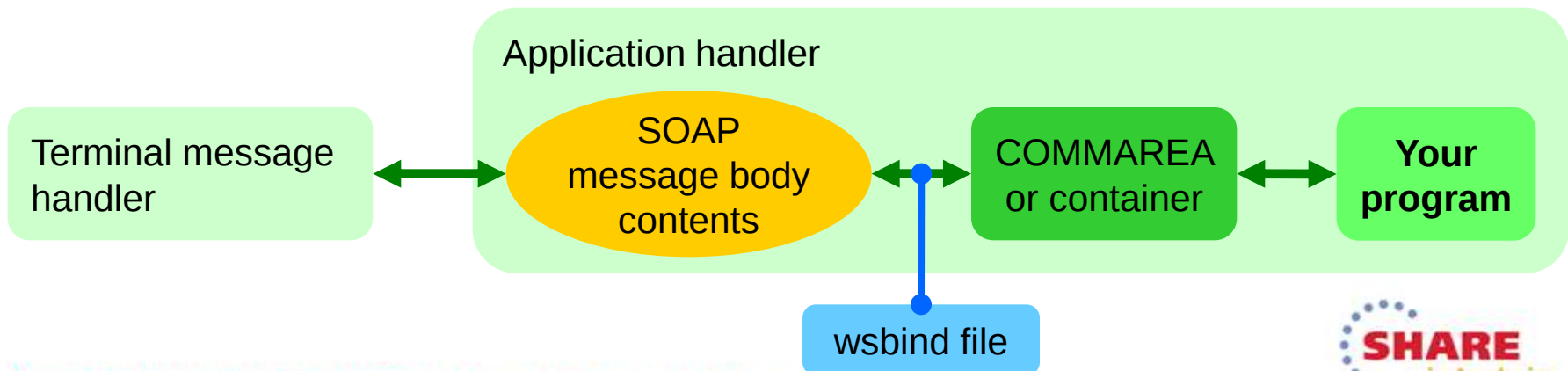
```
<provider_pipeline ... >
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

```
<requester_pipeline ... >
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler/>
    </service_handler_list>
  </service>
</requester_pipeline>
```

Also known as a “wrapper” program. Extracts data from XML, calls your CICS application program, converts returned data back into XML.

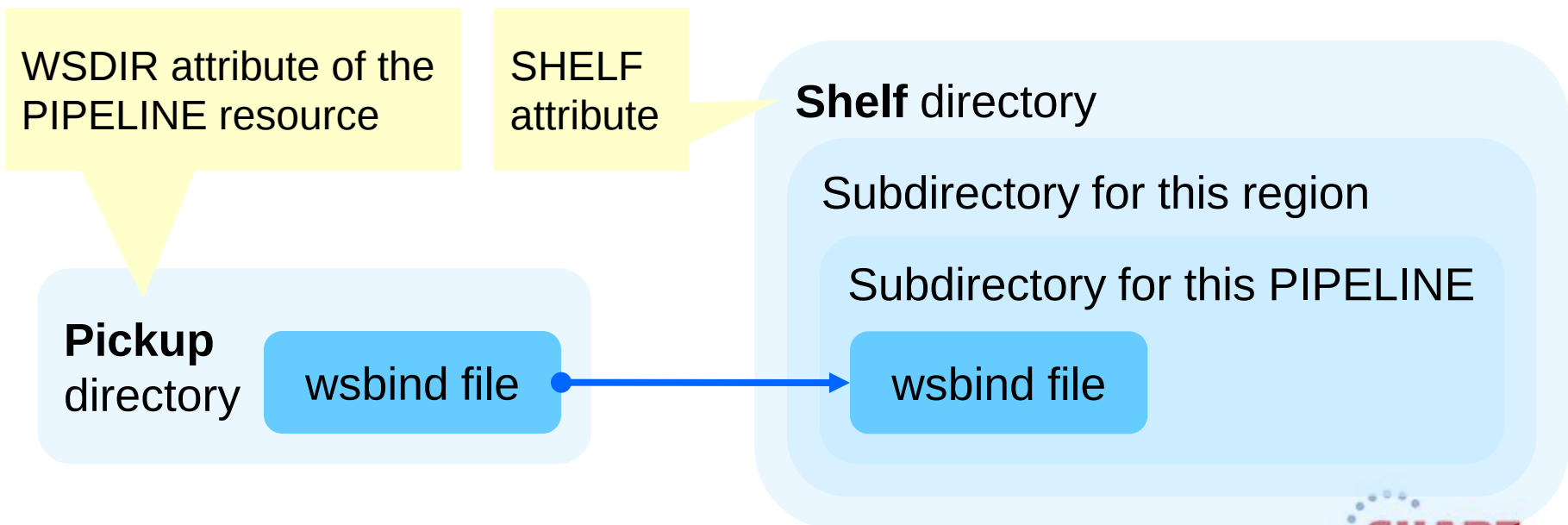
Web service binding (wsbind) file

- Generated by CICS web services assistant or RDz
- Proprietary to CICS web services
- Contains web service-specific information, such as how to map between the fields in a COMMAREA or container and the XML in a SOAP message body
- Enables you to use the CICS-supplied application handler (DFHPITP) for different web services

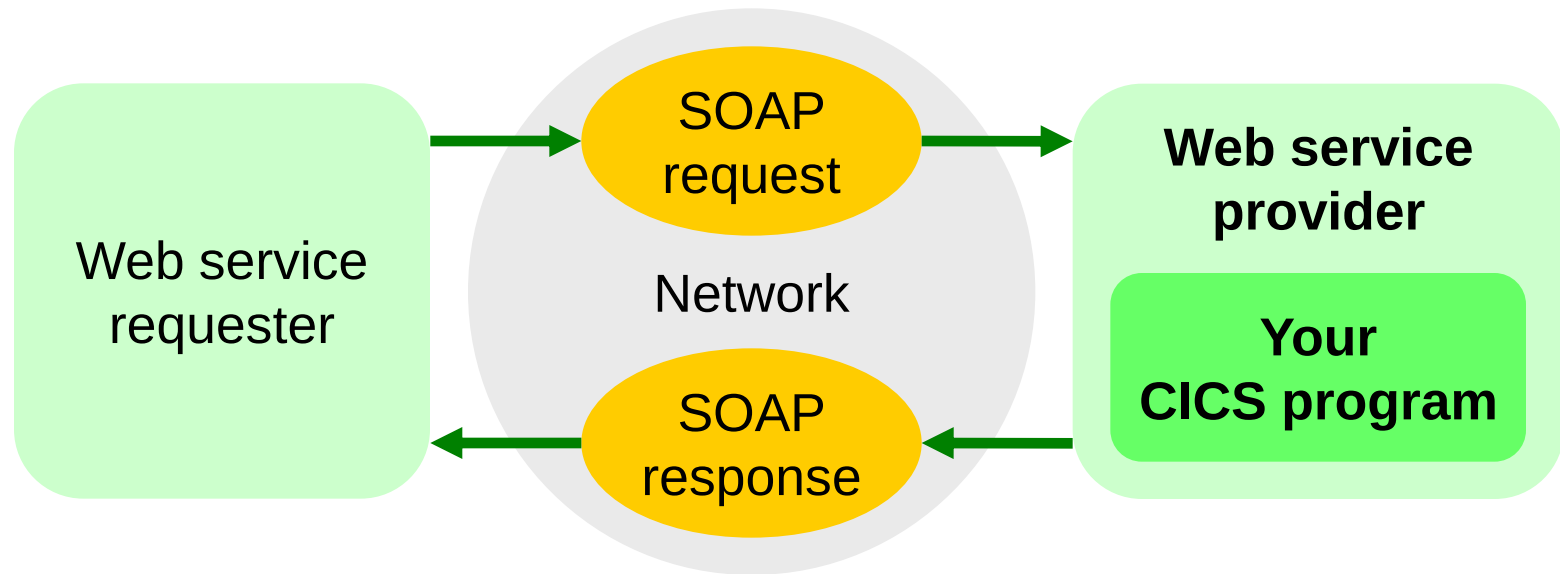


wsbind file: pickup and shelf directories

- When you install the PIPELINE resource, or when you issue a PIPELINE SCAN command, CICS copies the wsbind file from the pickup directory to the shelf directory.
- At runtime, CICS refers to the copy in the shelf directory.



Creating a web service provider in CICS

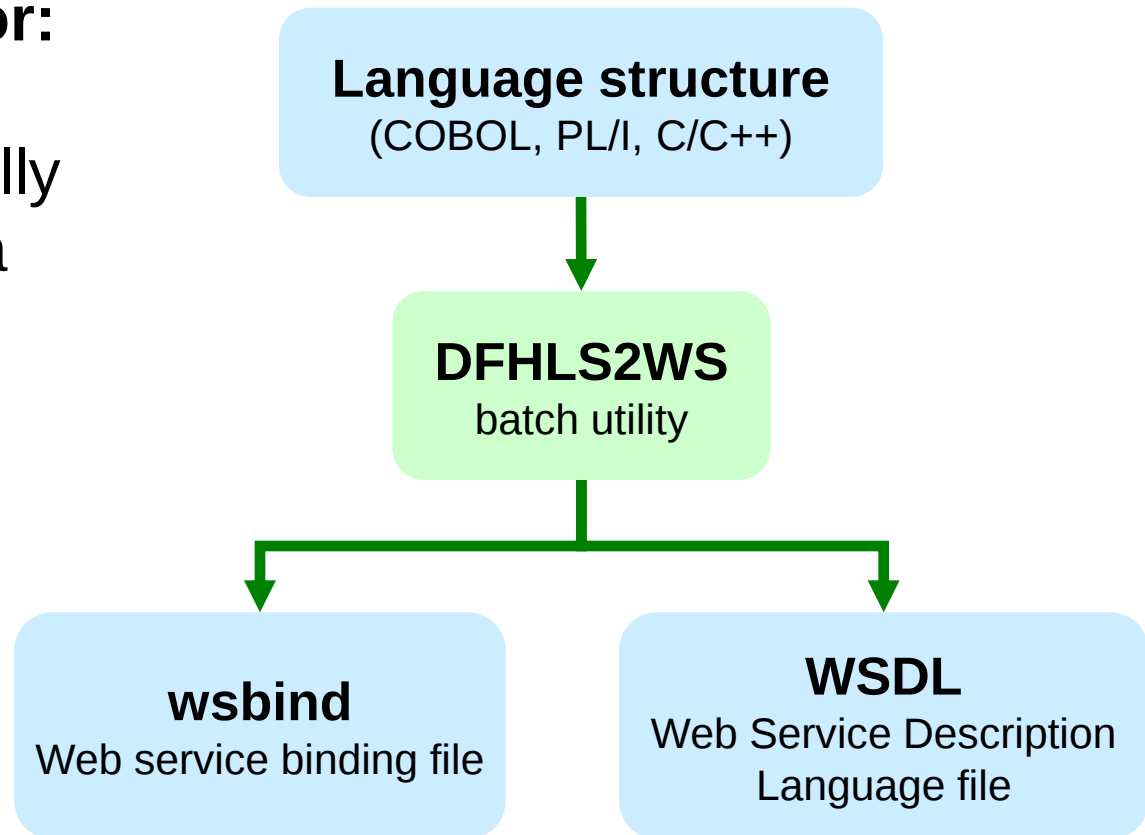


Methods for creating a web service provider in CICS

1. **CICS web services assistant** (batch utilities supplied with CICS) from a copybook, using the DFHLS2WS batch utility (generates a WSDL file and a wsbind file)
2. **Rational Developer for System z (RDz)** from a copybook (using a wizard), with *interpretive* runtime XML conversion (as per DFHLS2WS, above)
3. **RDz** as above, but with *compiled* runtime XML conversion (in addition to WSDL and wsbind files, also generates a bespoke COBOL program to convert XML)
4. **RDz Service Flow Modeler** from a recording of an interactive CICS terminal user interface (and using a wizard)

Creating a provider using the CICS web services assistant

- **Use this method for:** an existing CICS application that is fully functional and has a COMMAREA or channel interface
- **You will need:** a COBOL copybook (or PL/I, C/C++ equivalent)



Creating the CICS infrastructure for a provider



- These steps apply to any method for creating a provider.
 1. Create a **TCPIP SERVICE** resource.
 2. Create a **pipeline configuration file**.
 3. Create a **PIPELINE** resource.
 4. Unless you use autoinstalled PROGRAM definitions, create a **PROGRAM** resource for each program in the pipeline.

Creating a provider using the CICS web services assistant



1. Run the **DFHLS2WS** batch utility (for example, specifying a COBOL copybook as the input file).
2. Copy the generated **wsbind** file to the pickup directory (the z/OS UNIX path specified by the WSDIR attribute of the PIPELINE resource).
Optionally, copy the generated **WSDL** file to the same path (if you want to validate the SOAP messages).
3. Install the **PIPELINE** (dynamically creates the WEBSERVICE and URIMAP resources).

The provider is ready for testing.

JCL to run DFHLS2WS

```
//SYSEGXLS JOB (39248C,A,T),'LS2WS',
// MSGCLASS=A,NOTIFY=&SYSUID,REGION=0M
// SET QT='''
//WHERE SMA JCLLIB ORDER=CIRCLE.CICSWS.PROCLIB
//JAVAPROG EXEC DFHLS2WS,
// JAVADIR='Java601_64/J6.0.1_64',PATHPREF='/u',TMPDIR='/u/tmp',
// TMPFILE=&QT.&SYSUID.&QT,USSDIR='cicsts42'
//INPUT.SYSUT1 DD *
PDSLIB=CIRCLE.CICSWS.COPYLIB
REQMEM=PAYCOM1
RESPMEM=PAYCOM1
PGMINT=COMMAREA
MAPPING-LEVEL=3.0
MINIMUM-RUNTIME-LEVEL=CURRENT
LANG=COBOL
PGMNAME=PAYBUS
URI=/paybus1
WSBIND=/u/usr/lpp/cicsts/cicsts42/samples/webservices/wsbinding/provider/p*
aybus1.wsbinding
WSDL=/u/usr/lpp/cicsts/cicsts42/samples/webservices/wsd1/paybus1.wsd1
LOGFILE=/u/sysegx0/paybus
/*
```

Input COBOL copybook PDS members:
one for the request, another for the
response (same in this case)

Output wsbind and
WSDL files

Your existing CICS program

DFHLS2WS log

DFHPI9609I Parameter "LOGFILE" has value "/u/sysegx0/paybus".

...

DFHPI9609I Parameter "PDSLIB" has value "//CIRCLE.CICSWS.COPYLIB".

DFHPI9609I Parameter "PGMINT" has value "COMMAREA".

DFHPI9609I Parameter "PGMNAME" has value "PAYBUS".

DFHPI9609I Parameter "REQMEM" has value "PAYCOM1".

...

DFHPI9609I Parameter "RESPMEM" has value "PAYCOM1".

...

DFHPI9609I Parameter "URI" has value "/paybus1".

...

DFHPI9629I The minimum runtime level required for this Web service is "3.0".

DFHPI9640I This Web service should be installed into a PIPELINE that uses SOAP version "1.1".

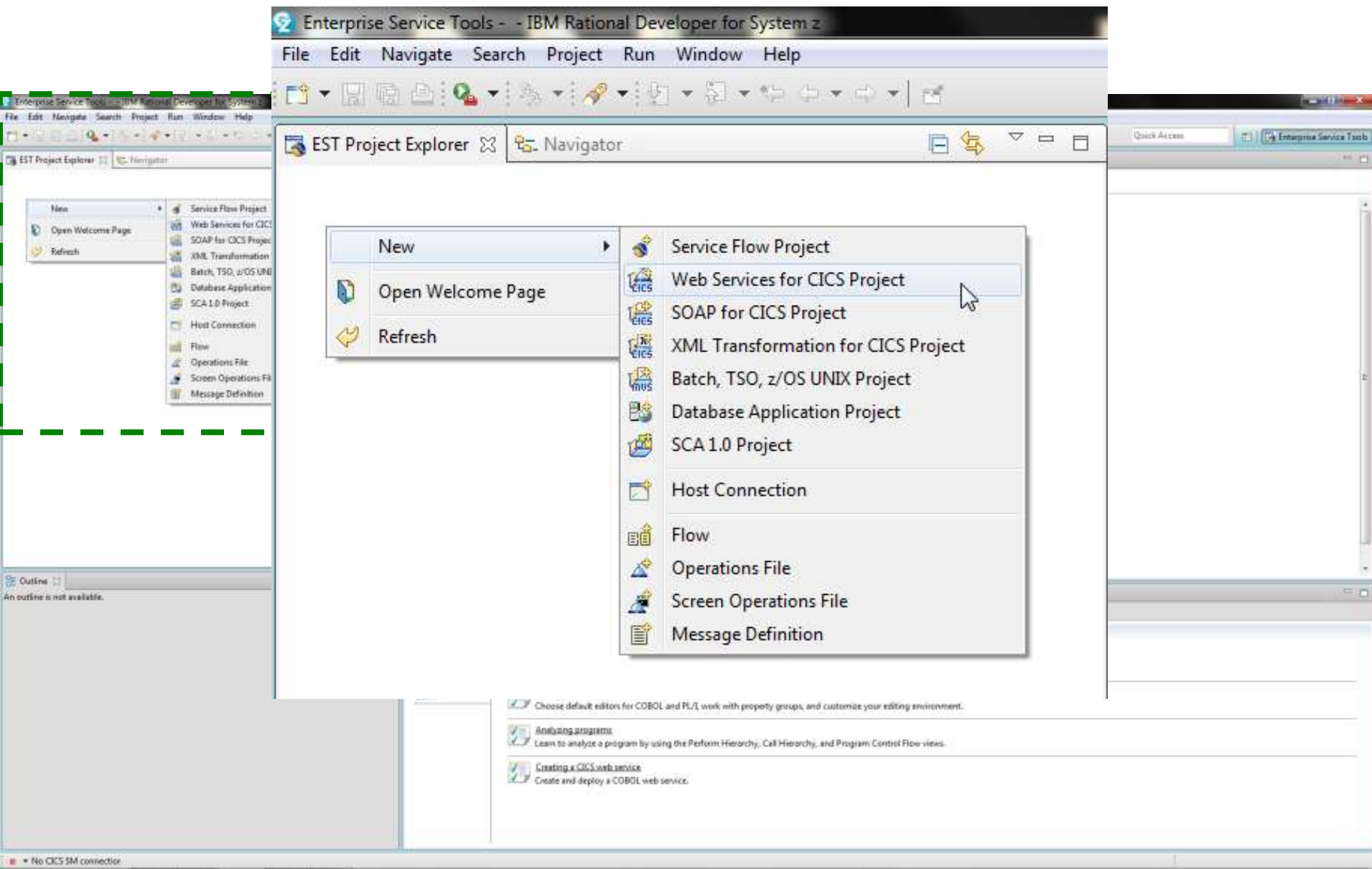
DFHPI9587I Program "DFHLS2WS" has completed SUCCESSFULLY.

Testing the provider using RDz Web Services Tester



- The following slides demonstrate using the RDz Web Services Tester to test the provider:
 1. Create a CICS web service project in RDz
 2. Import the WSDL file
 3. Run the Web Services Tester
 4. Use the GUI to create and send a request to the provider

Testing the provider using RDz (1 of 8)



The screenshot displays the IBM Rational Developer for System z (RDz) Enterprise Service Tools interface. The main window is titled "Enterprise Service Tools - IBM Rational Developer for System z". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains icons for file operations and development tasks.

The "EST Project Explorer" and "Navigator" views are visible. The "New" menu is open, showing the following options:

- New
- Open Welcome Page
- Refresh

The "New" submenu is expanded, showing the following project types:

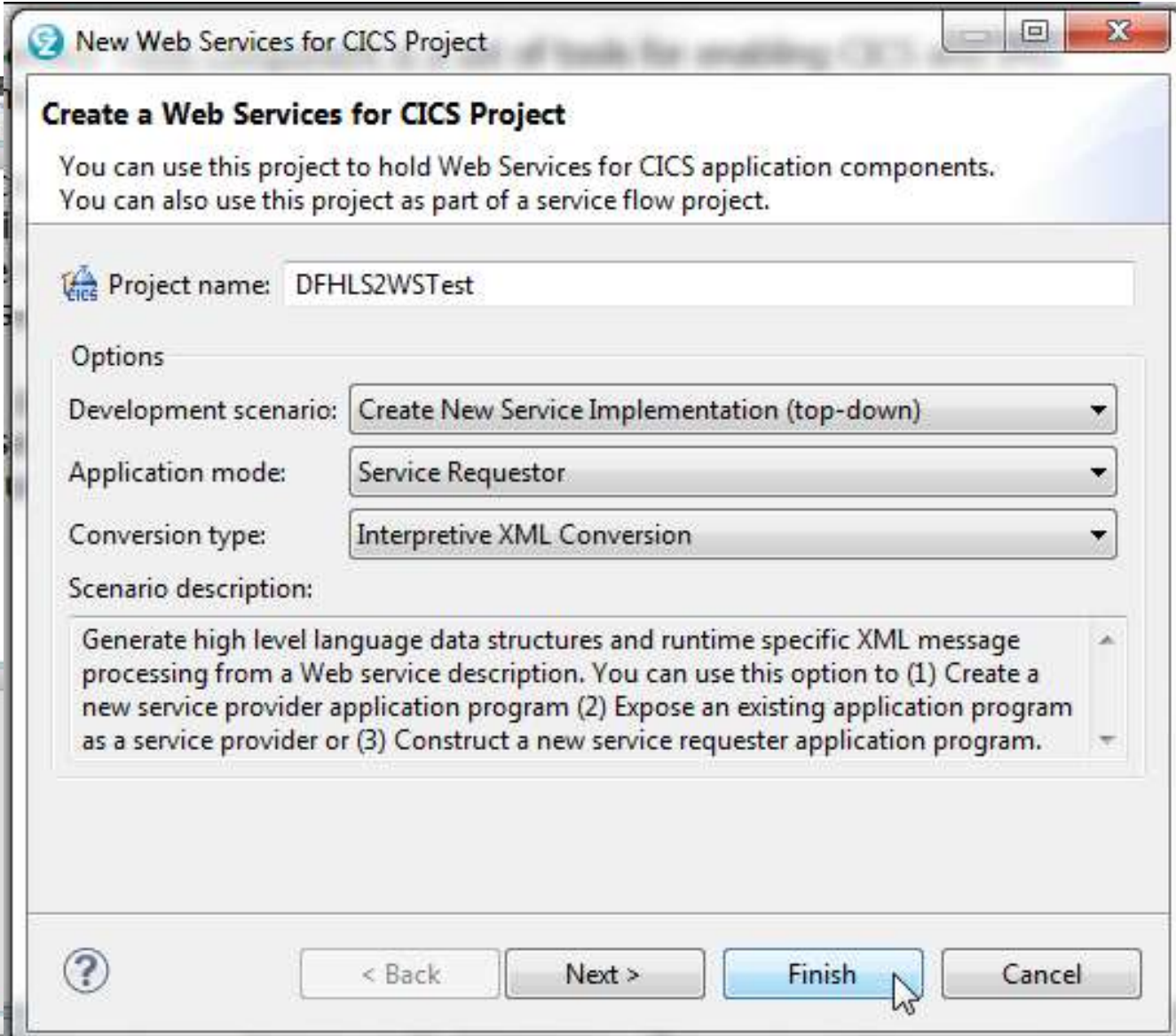
- Service Flow Project
- Web Services for CICS Project (highlighted)
- SOAP for CICS Project
- XML Transformation for CICS Project
- Batch, TSO, z/OS UNIX Project
- Database Application Project
- SCA 1.0 Project
- Host Connection
- Flow
- Operations File
- Screen Operations File
- Message Definition

The "Outline" view at the bottom left shows "An outline is not available." The "Task List" view at the bottom right shows the following tasks:

- Choose default editors for COBOL and PL/I, work with property groups, and customize your editing environment.
- Analyzing programs: Learn to analyze a program by using the Perform Hierarchy, Call Hierarchy, and Program Control Flow views.
- Creating a CICS web service: Create and deploy a COBOL web service.

The status bar at the bottom left indicates "No CICS SM connection."

Testing the provider using RDz (2 of 8)



Create a Web Services for CICS Project

You can use this project to hold Web Services for CICS application components.
You can also use this project as part of a service flow project.

 Project name: DFHLS2WSTest

Options

Development scenario: Create New Service Implementation (top-down) ▼

Application mode: Service Requestor ▼

Conversion type: Interpretive XML Conversion ▼

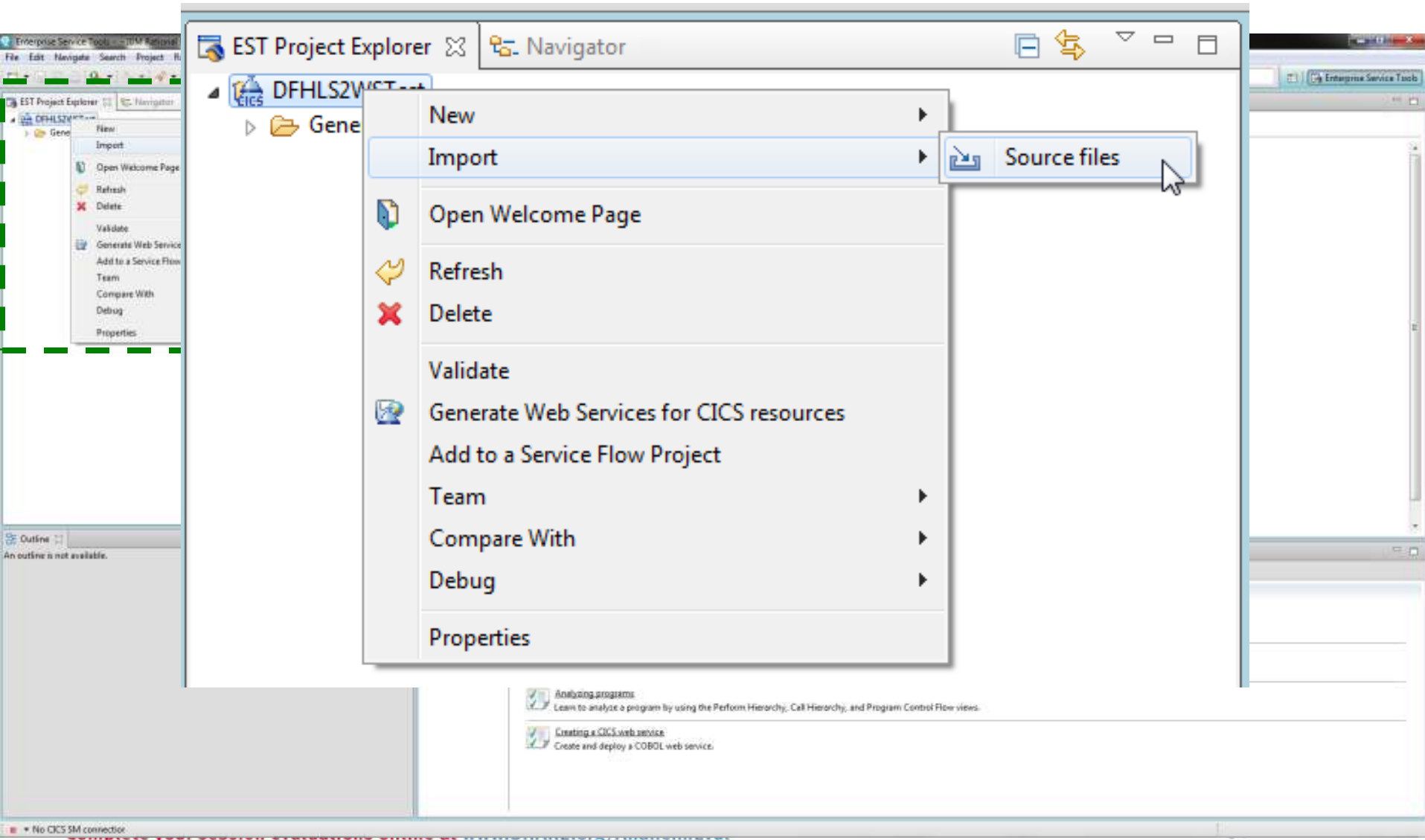
Scenario description:

Generate high level language data structures and runtime specific XML message processing from a Web service description. You can use this option to (1) Create a new service provider application program (2) Expose an existing application program as a service provider or (3) Construct a new service requester application program.

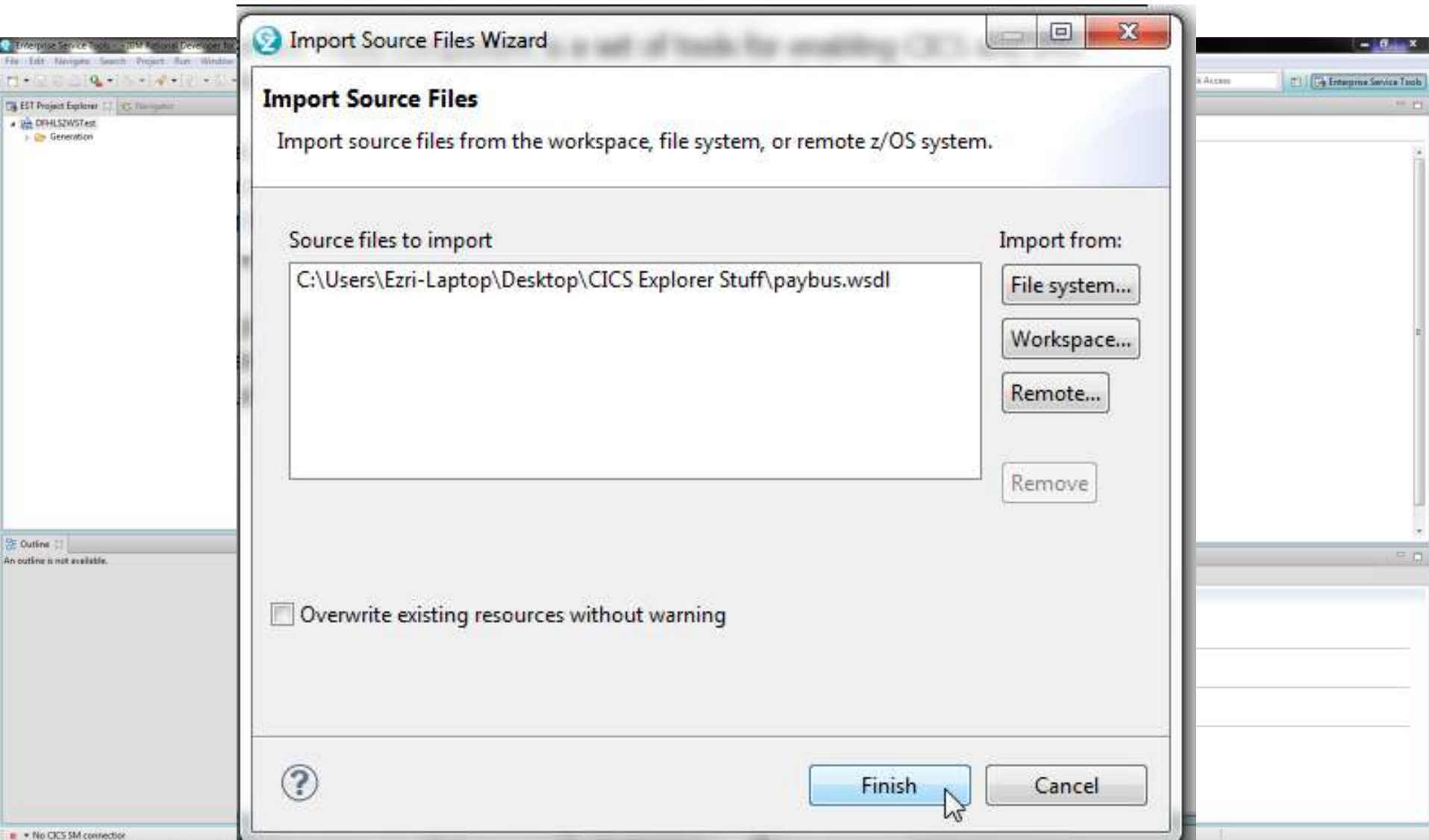
 < Back Next > **Finish** Cancel

• No CICS SM connection

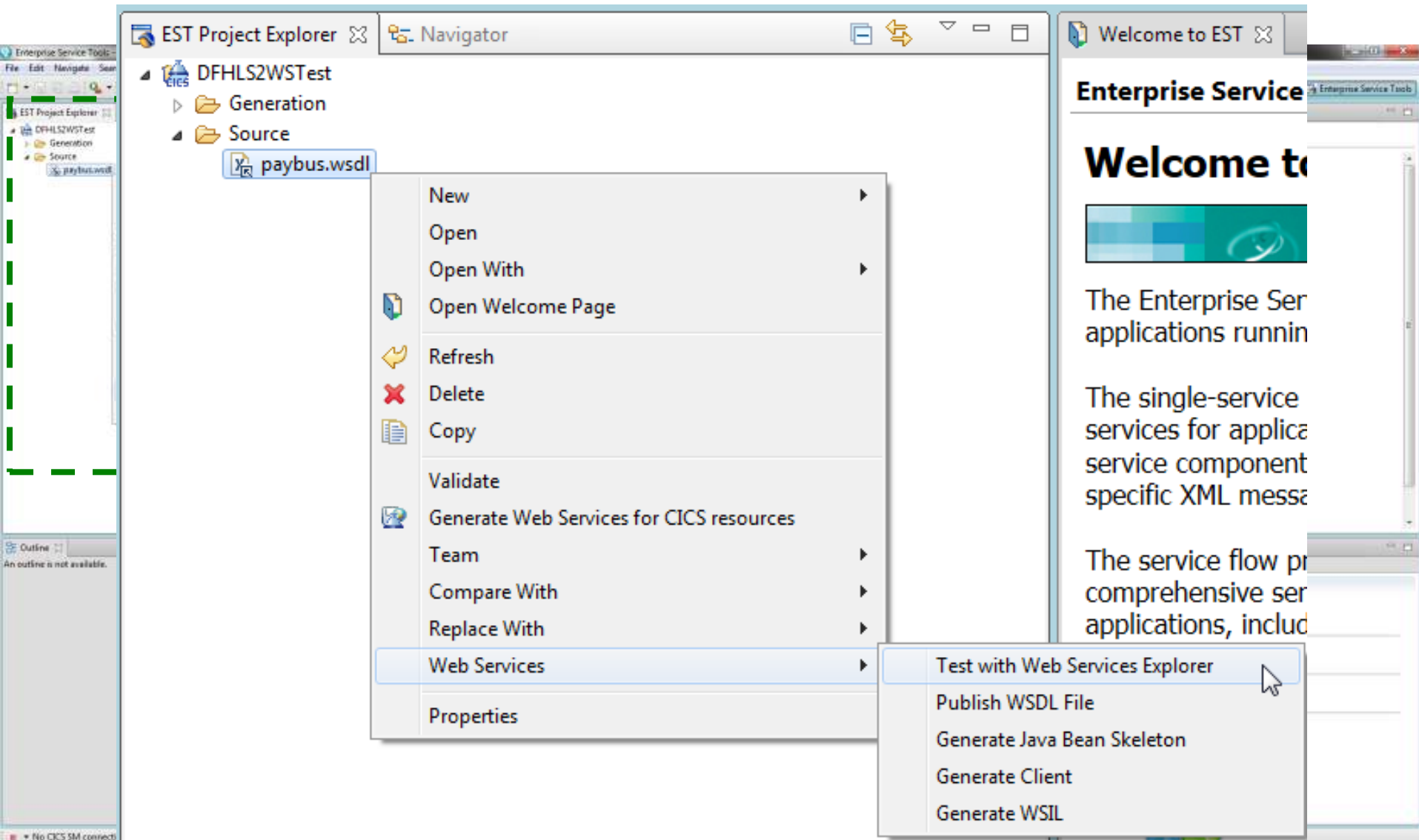
Testing the provider using RDz (3 of 8)



Testing the provider using RDz (4 of 8)

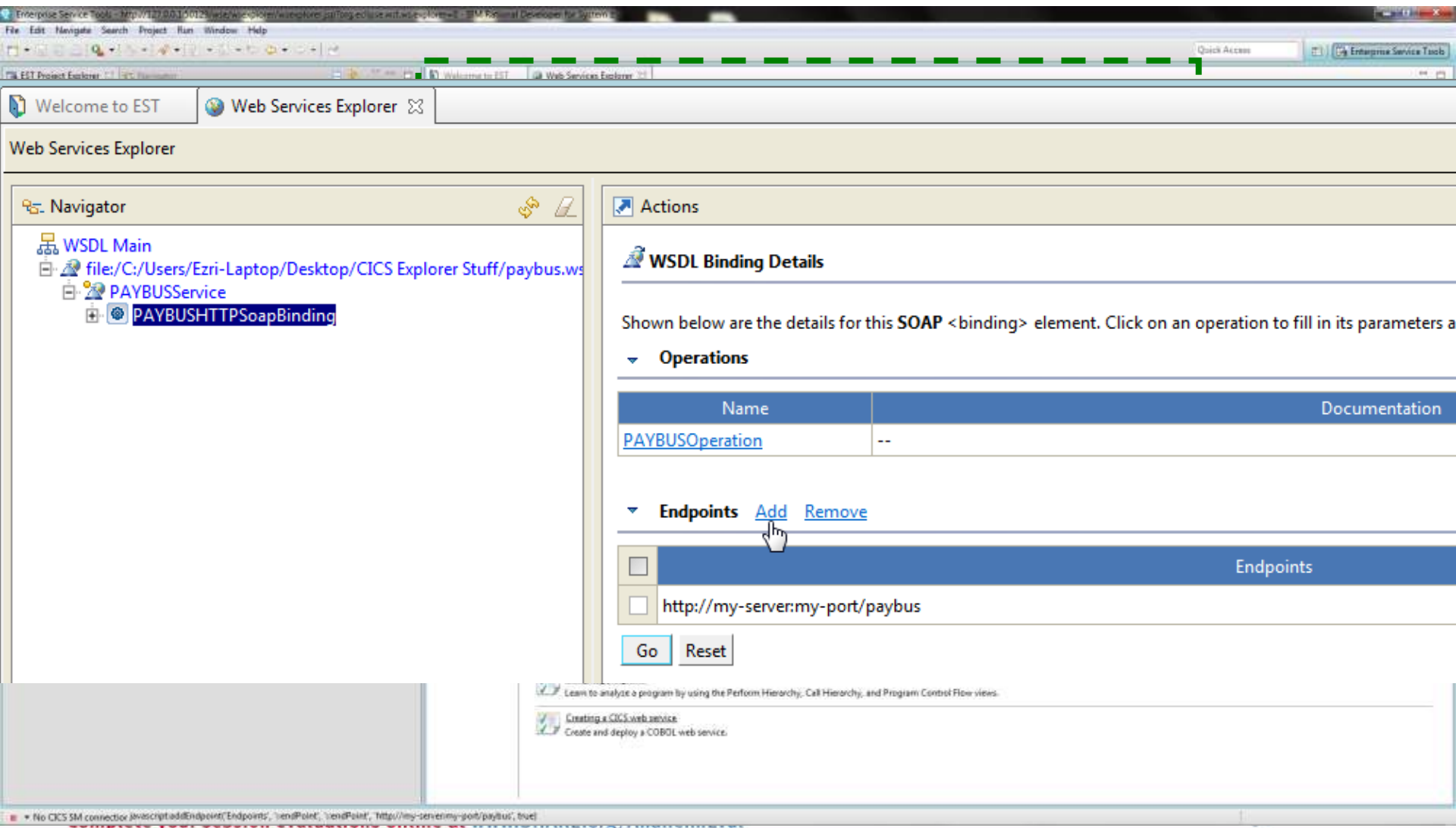


Testing the provider using RDz (5 of 8)



The screenshot displays the IBM Enterprise Service Tools (EST) interface. The 'EST Project Explorer' on the left shows a project named 'DFHLS2WSTest' with a 'Source' folder containing 'paybus.wsdl'. A right-click context menu is open over 'paybus.wsdl', listing options such as 'New', 'Open', 'Open With', 'Open Welcome Page', 'Refresh', 'Delete', 'Copy', 'Validate', 'Generate Web Services for CICS resources', 'Team', 'Compare With', 'Replace With', 'Web Services', and 'Properties'. The 'Web Services' option is selected, and a sub-menu is visible with the following actions: 'Test with Web Services Explorer', 'Publish WSDL File', 'Generate Java Bean Skeleton', 'Generate Client', and 'Generate WSIL'. The 'Test with Web Services Explorer' option is highlighted by the mouse cursor. In the background, the 'Welcome to EST' dialog is partially visible, featuring the text: 'Enterprise Service Tools', 'Welcome to', 'The Enterprise Service Tools applications running...', 'The single-service services for application service component specific XML messages', and 'The service flow provides comprehensive service applications, including...'. The bottom status bar indicates 'No CICS SM connect'.

Testing the provider using RDz (6 of 8)



The screenshot shows the Enterprise Service Tools (EST) Web Services Explorer interface. The left pane displays the project structure with the following items:

- WSDL Main
 - file:/C:/Users/Ezri-Laptop/Desktop/CICS Explorer Stuff/paybus.ws
 - PAYBUSService
 - PAYBUSHTTPSoapBinding

The right pane shows the **WSDL Binding Details** for the selected binding. It includes the following sections:

- Operations**: A table listing the available operations.
- Endpoints**: A section for managing endpoints, including an **Add** button and a list of existing endpoints.

Name	Documentation
PAYBUSOperation	--

Endpoints	
☐	
☐	http://my-server:my-port/paybus

Buttons: **Go** **Reset**

At the bottom, there are two informational messages:

- Learn to analyze a program by using the Perform Hierarchy, Call Hierarchy, and Program Control Flow views.
- Creating a CICS web service: Create and deploy a COBOL web service.

The status bar at the bottom indicates: No CICS SM connectio javascript:addEndpoint('Endpoint', 'endPoint', 'http://my-server:my-port/paybus', true)

Testing the provider using RDz (7 of 8)

Welcome to EST Web Services Explorer

Web Services Explorer

Navigator

- WSDL Main
 - file:/C:/Users/Ezri-Laptop/Desktop/CICS Explorer Stuff/paybus.ws
 - PAYBUSService
 - PAYBUSHTTPSoapBinding

Actions

WSDL Binding Details

Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and

Operations

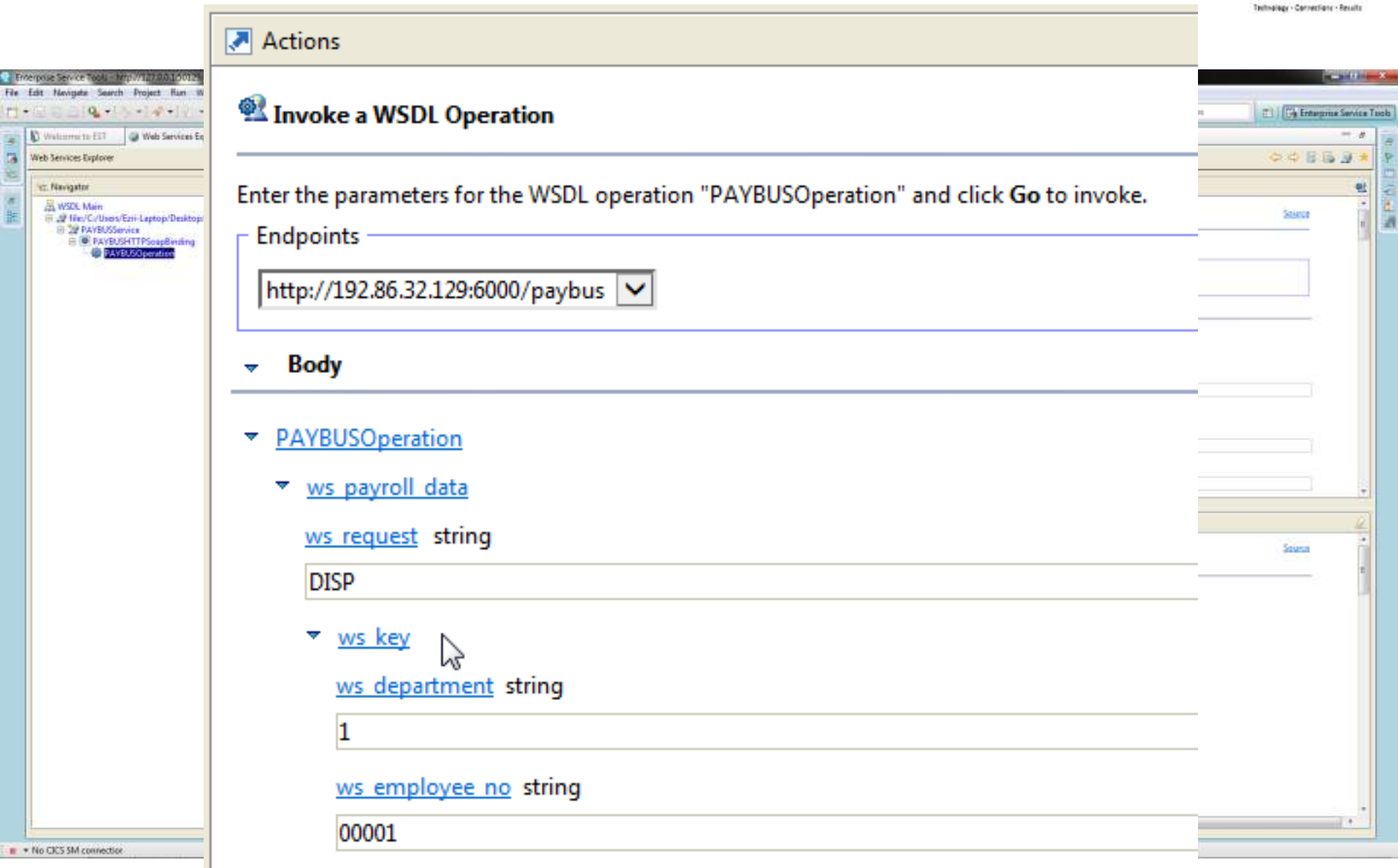
Name	Documentation
PAYBUSOperation	--

Endpoints [Add](#) [Remove](#)

	Endpoints
☐	
☐	http://my-server:my-port/paybus
☒	http://192.86.32.129:6000/paybus

[Go](#) [Reset](#)

Testing the provider using RDz (8 of 8)



The screenshot shows the Enterprise Service Tools (EST) interface. On the left, the 'Web Services Explorer' pane displays a tree view with 'WSDL Main' expanded, showing 'PAYBUSOperation' selected. The main area is titled 'Actions' and contains the 'Invoke a WSDL Operation' dialog. The dialog has a text box for 'Endpoints' containing 'http://192.86.32.129:6000/paybus'. Below this, the 'Body' section is expanded, showing the 'PAYBUSOperation' operation. Under 'PAYBUSOperation', the 'ws payroll data' section is expanded, showing 'ws request' with a value of 'DISP'. The 'ws key' section is also expanded, showing 'ws department' with a value of '1'. The 'ws employee no' section is expanded, showing a value of '00001'. A mouse cursor is hovering over the 'ws key' section. The bottom status bar indicates 'No ICES SM connection'.

Actions

Invoke a WSDL Operation

Enter the parameters for the WSDL operation "PAYBUSOperation" and click **Go** to invoke.

Endpoints

http://192.86.32.129:6000/paybus

Body

PAYBUSOperation

ws payroll data

ws request string

DISP

ws key

ws department string

1

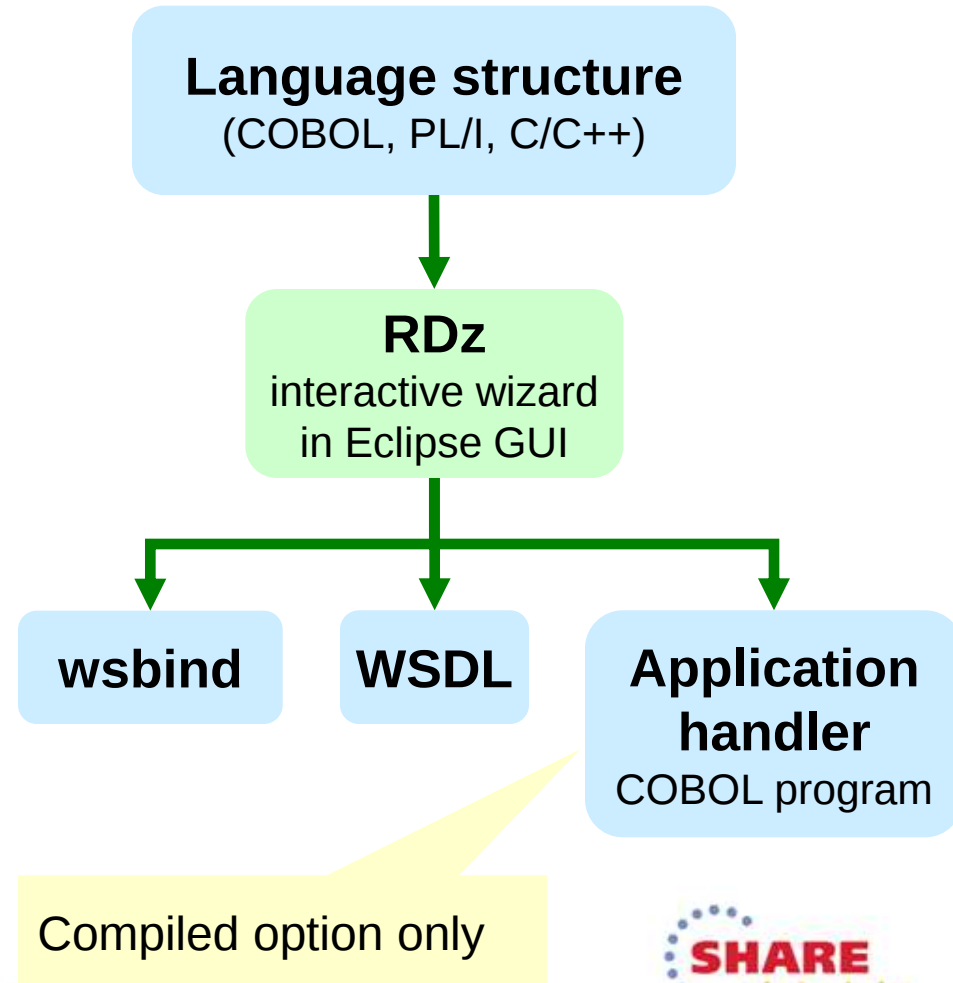
ws employee no string

00001

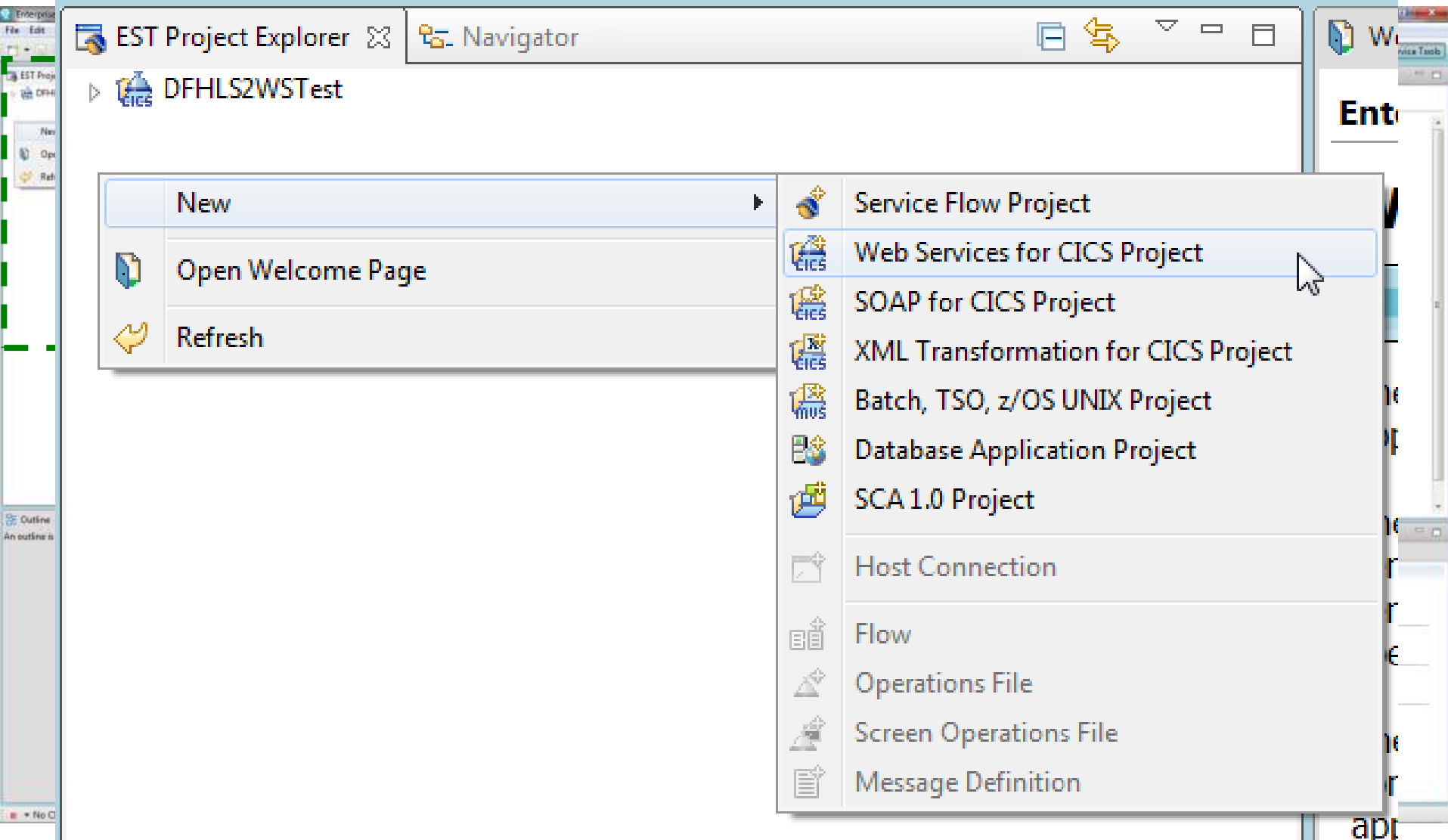
No ICES SM connection

Creating a provider using Rational Developer for System z (RDz)

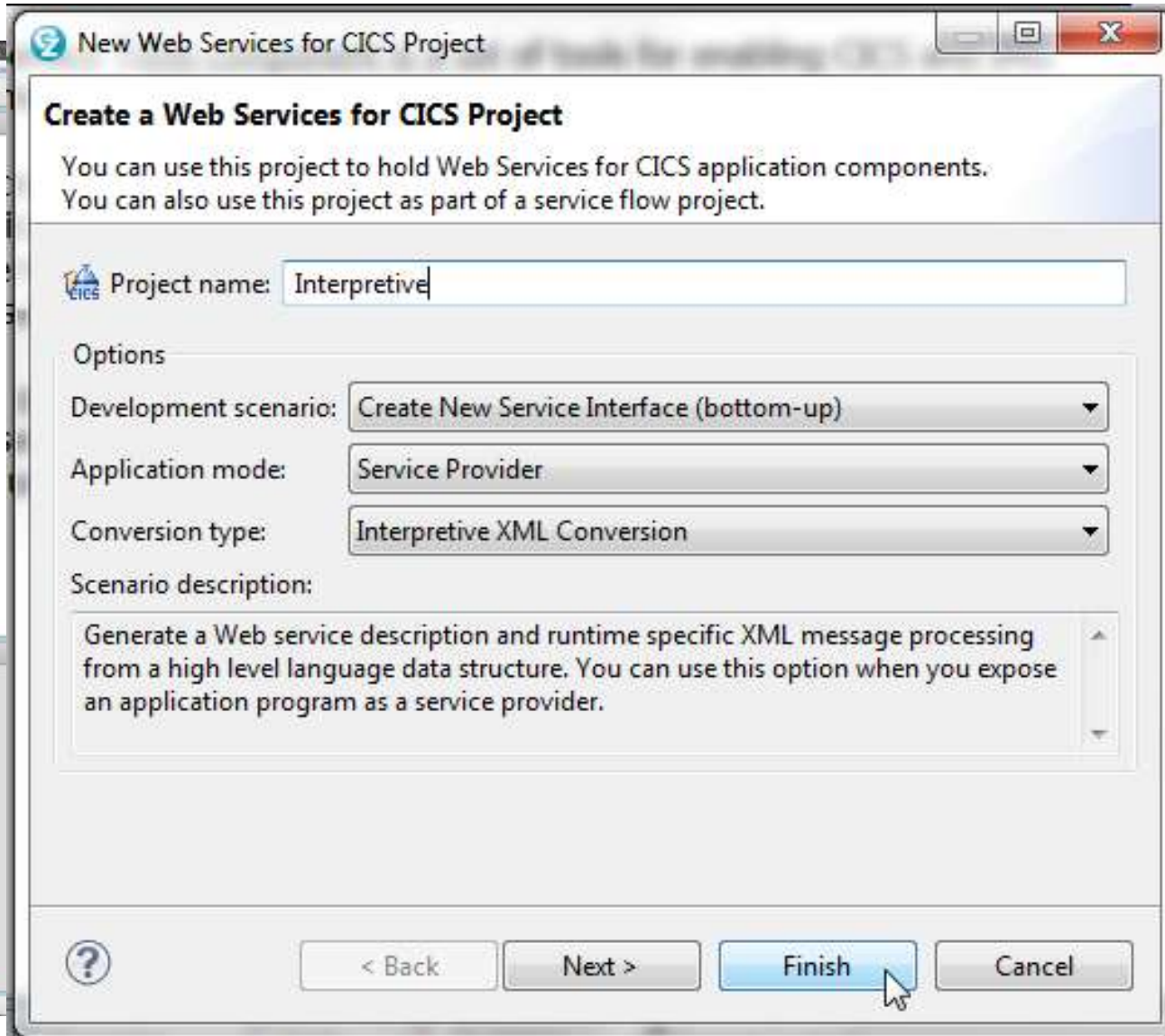
- Step-by-step wizard, with two options for runtime XML conversion:
- **Interpretive** uses a standard wrapper program, as per the CICS assistant
- **Compiled** generates a bespoke COBOL application handler (wrapper program)



Creating a provider using RDz: interpretive (1 of 9)



Creating a provider using RDz: interpretive (2 of 9)



New Web Services for CICS Project

Create a Web Services for CICS Project

You can use this project to hold Web Services for CICS application components.
You can also use this project as part of a service flow project.

 Project name:

Options

Development scenario:

Application mode:

Conversion type:

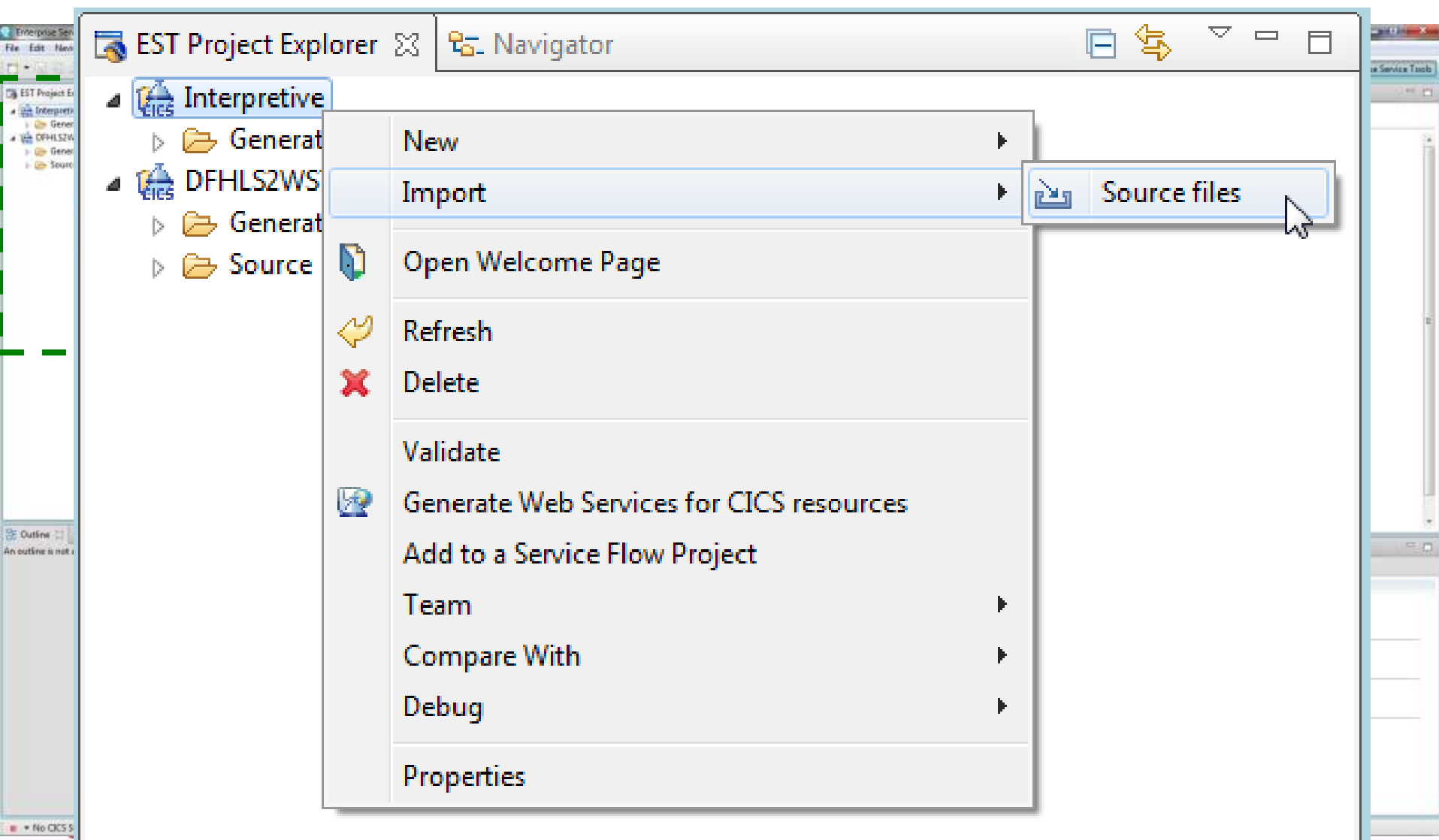
Scenario description:

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

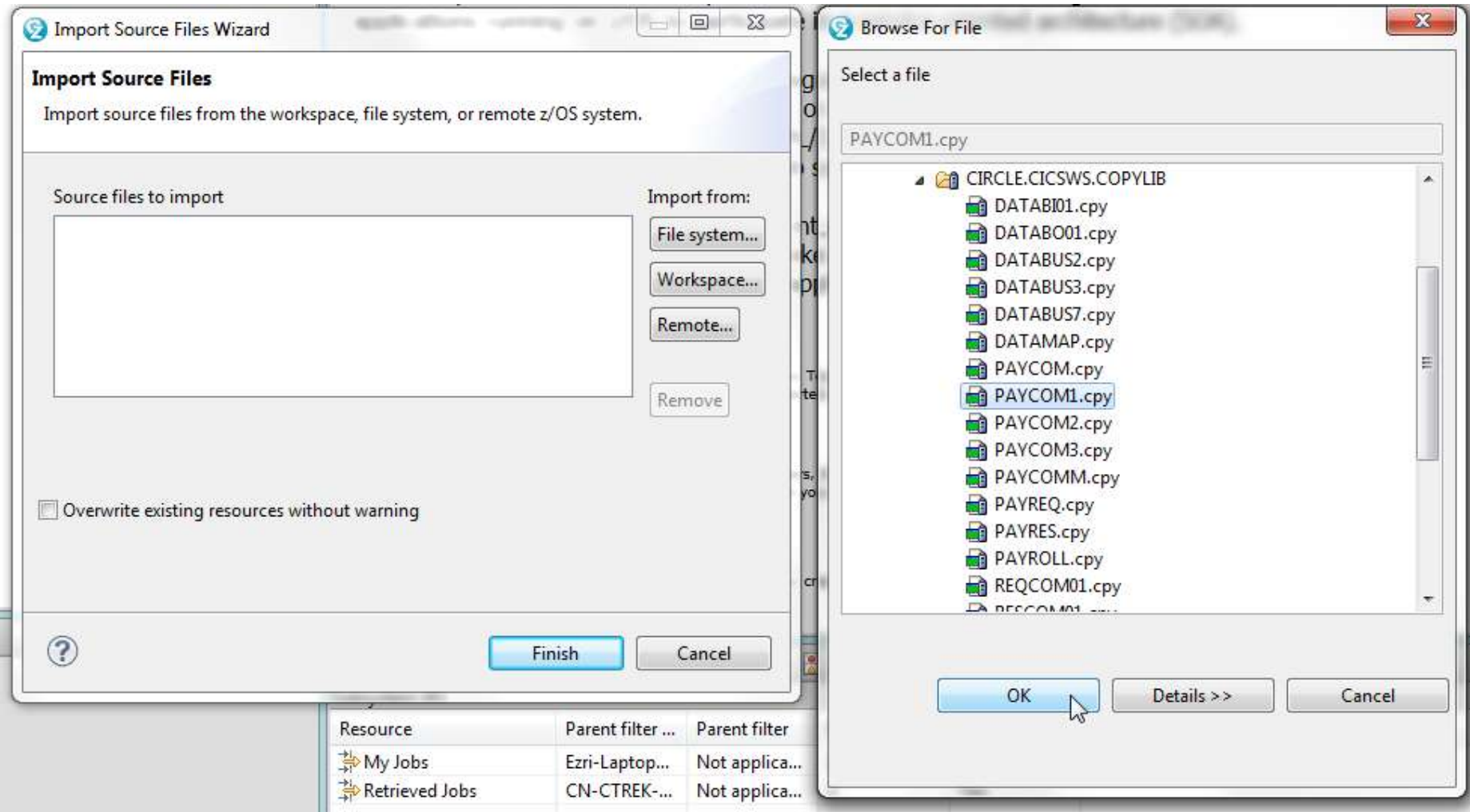


No CICS SM connection

Creating a provider using RDz: interpretive (3 of 9)



Creating a provider using RDz: interpretive (4 of 9)



The screenshot shows two overlapping windows from a software application. The 'Import Source Files Wizard' window is in the foreground, and the 'Browse For File' window is open behind it.

Import Source Files Wizard

Import Source Files
Import source files from the workspace, file system, or remote z/OS system.

Source files to import

Import from:

- File system...
- Workspace...
- Remote...
- Remove

☐ Overwrite existing resources without warning

Finish Cancel

Browse For File

Select a file

PAYCOM1.cpy

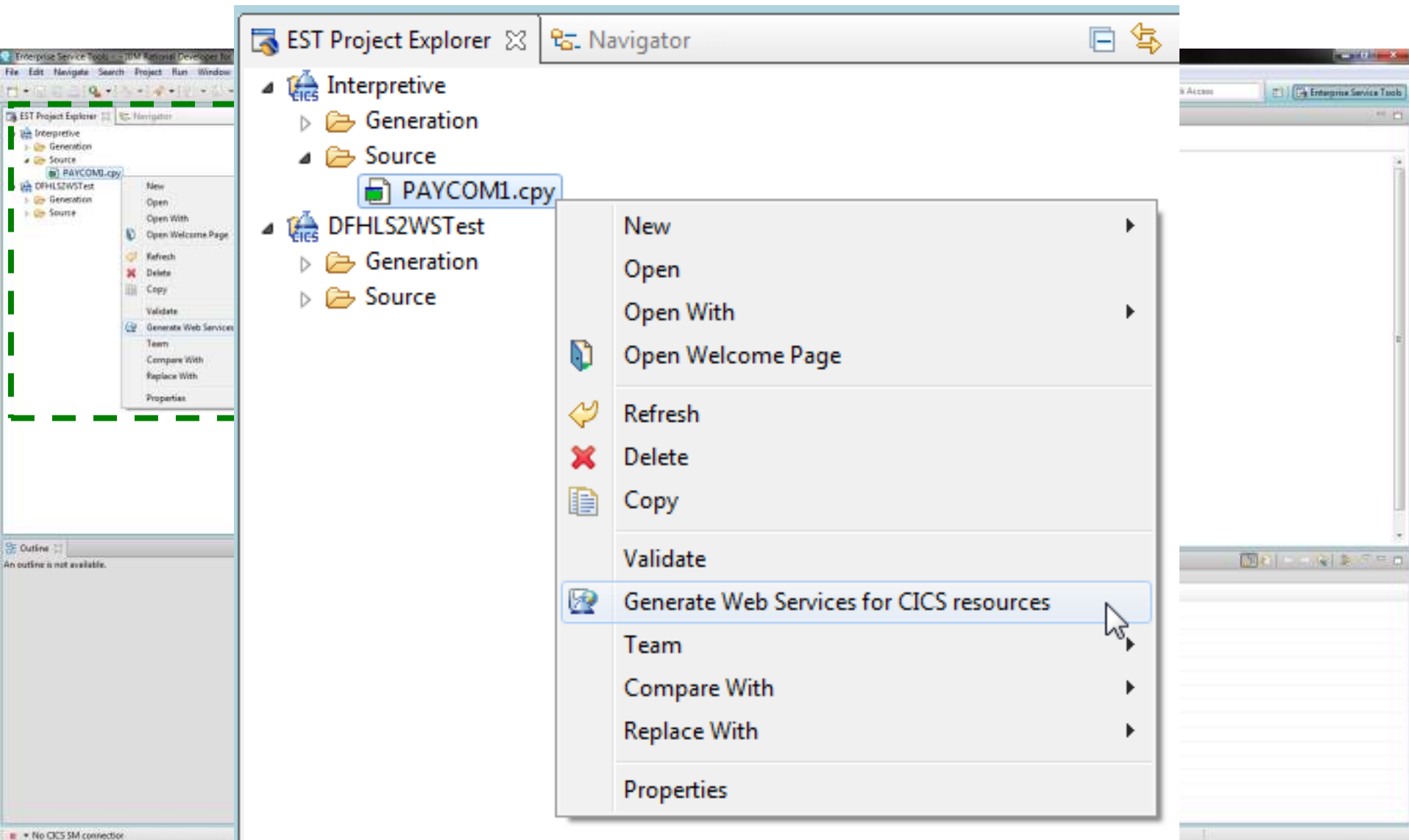
CIRCLE.CICSWS.COPYLIB

- DATABI01.cpy
- DATABO01.cpy
- DATABUS2.cpy
- DATABUS3.cpy
- DATABUS7.cpy
- DATAMAP.cpy
- PAYCOM.cpy
- PAYCOM1.cpy**
- PAYCOM2.cpy
- PAYCOM3.cpy
- PAYCOMM.cpy
- PAYREQ.cpy
- PAYRES.cpy
- PAYROLL.cpy
- REQCOM01.cpy
- REQCOM01.cpy

OK Details >> Cancel

Resource	Parent filter ...	Parent filter
My Jobs	Ezri-Laptop...	Not applica...
Retrieved Jobs	CN-CTREK-...	Not applica...

Creating a provider using RDz: interpretive (5 of 9)



The screenshot displays the IBM Rational Developer for iSeries interface. The **EST Project Explorer** pane on the left shows the project structure:

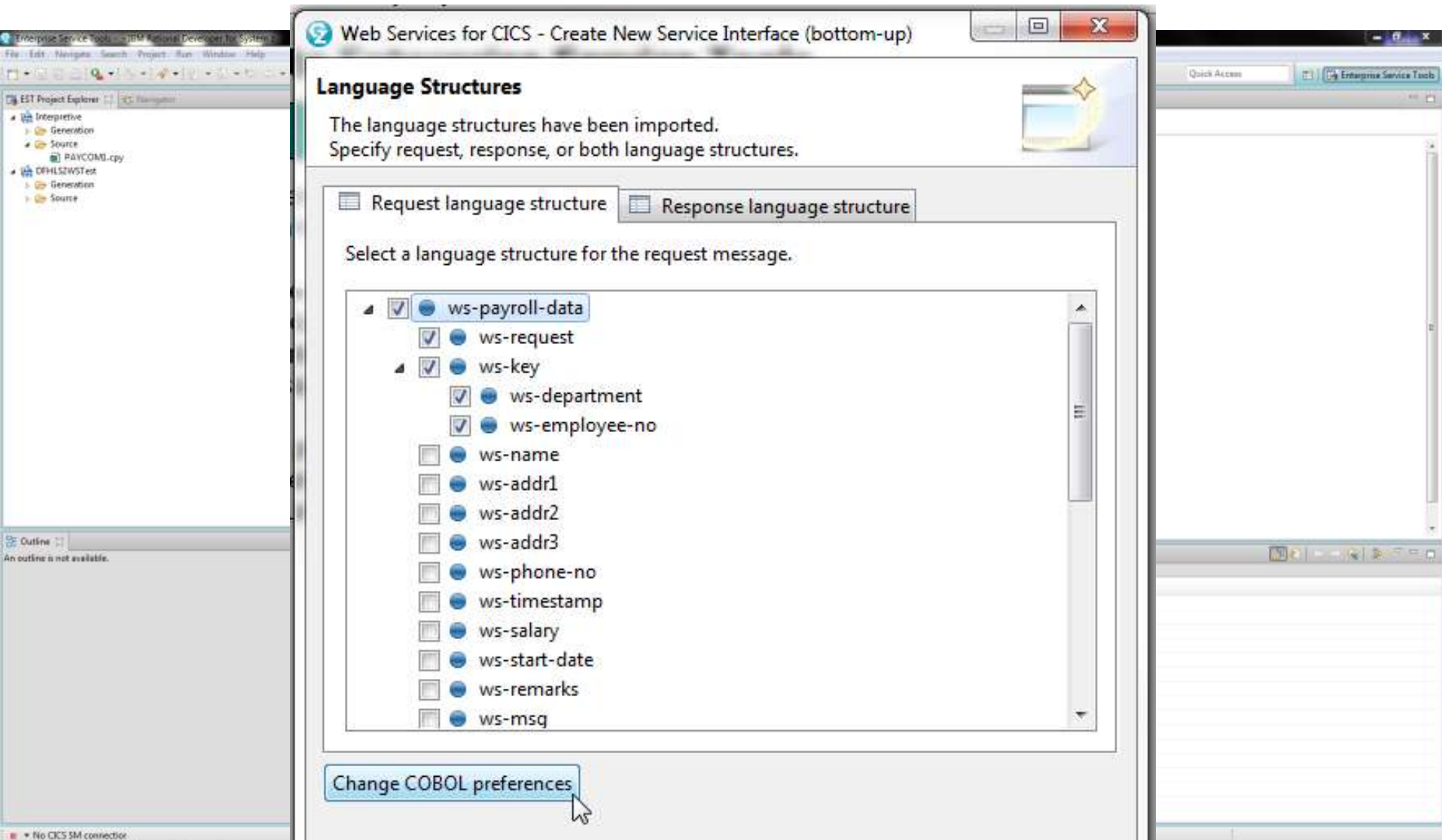
- Interpretive
 - Generation
 - Source
 - PAYCOM1.cpy**
- DFHLS2WSTest
 - Generation
 - Source

A right-click context menu is open for the **PAYCOM1.cpy** file. The menu options are:

- New
- Open
- Open With
- Open Welcome Page
- Refresh
- Delete
- Copy
- Validate
- Generate Web Services for CICS resources** (highlighted by the mouse)
- Team
- Compare With
- Replace With
- Properties

The **Outline** pane at the bottom left indicates "An outline is not available." The status bar at the bottom shows "No CICS SM connection."

Creating a provider using RDz: interpretive (6 of 9)



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

Language Structures

The language structures have been imported.
Specify request, response, or both language structures.

☒ Request language structure ☐ Response language structure

Select a language structure for the request message.

- ☒ ws-payroll-data
 - ☒ ws-request
 - ☒ ws-key
 - ☒ ws-department
 - ☒ ws-employee-no
 - ☐ ws-name
 - ☐ ws-addr1
 - ☐ ws-addr2
 - ☐ ws-addr3
 - ☐ ws-phone-no
 - ☐ ws-timestamp
 - ☐ ws-salary
 - ☐ ws-start-date
 - ☐ ws-remarks
 - ☐ ws-msg

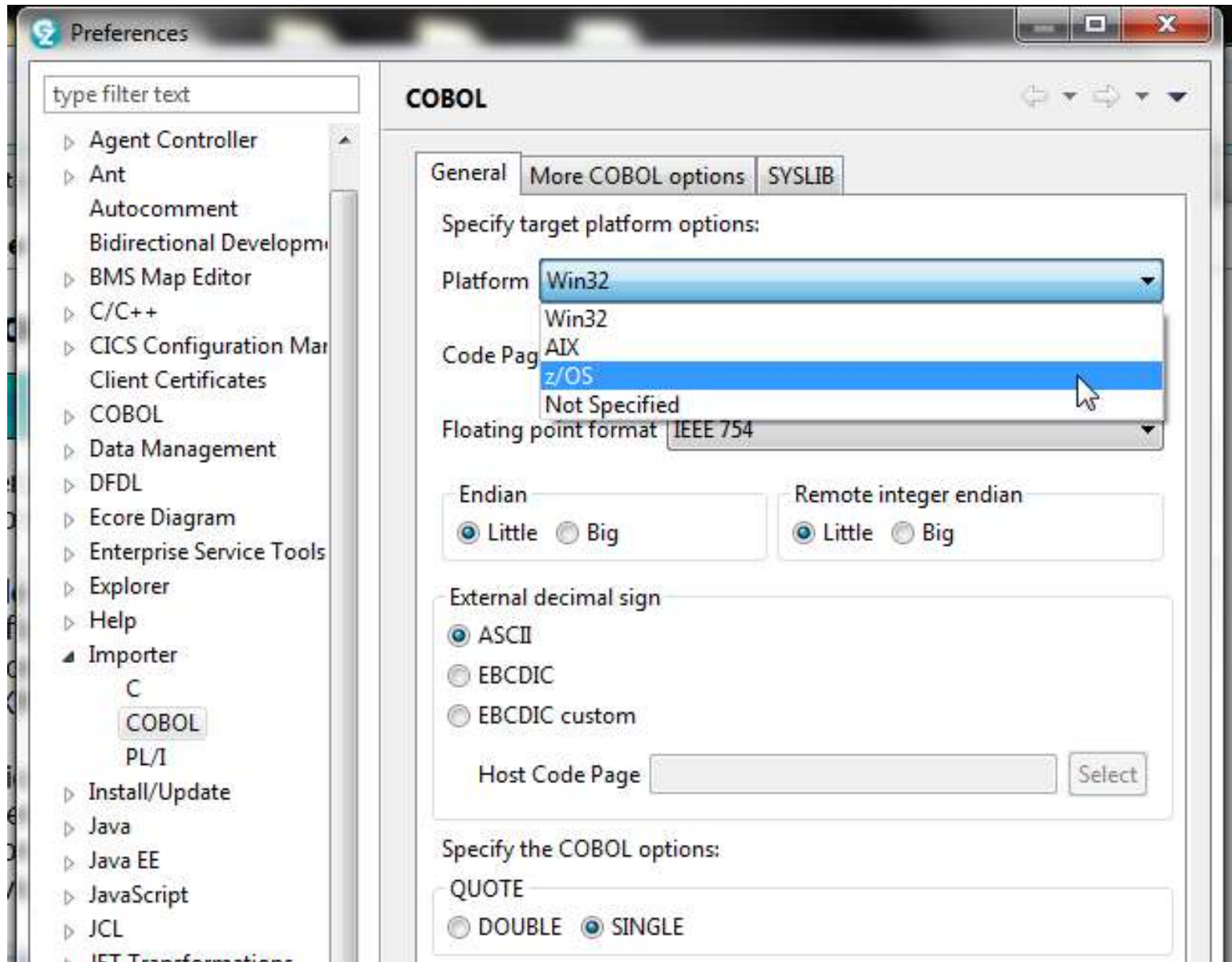
[Change COBOL preferences](#)

EST Project Explorer: Interpretive, Generation, Source, PAYCOMB.cpy, DFHLS2WST est, Generation, Source

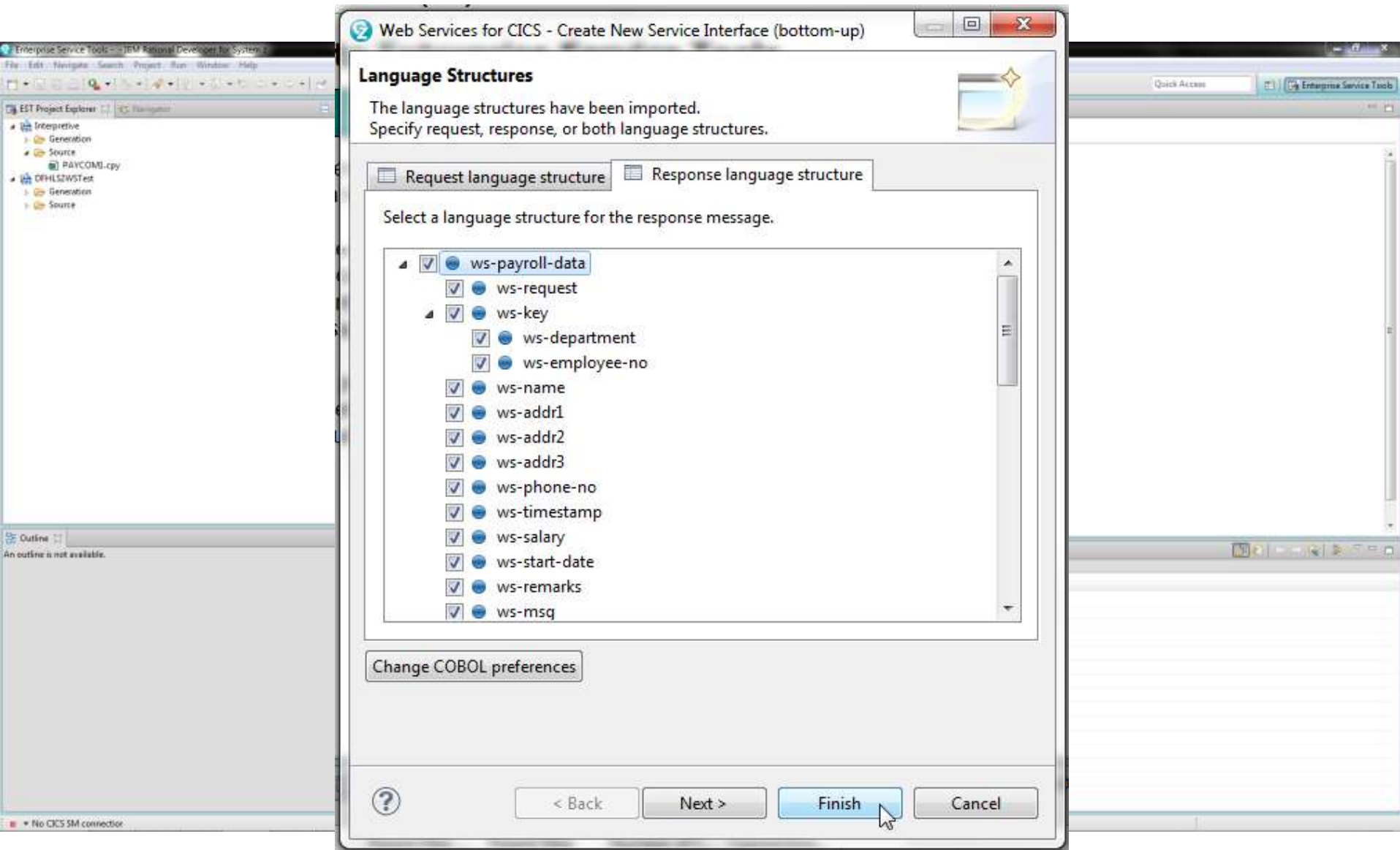
Outline: An outline is not available.

No CICS SM connection

Creating a provider using RDz: interpretive (7 of 9)



Creating a provider using RDz: interpretive (8 of 9)



Creating a provider using RDz: interpretive (9 of 9)



Enterprise Service Tools - InterpretiveGenerator - PAYCOM1.wsbind - IBM Rational Developer for System z

Welcome to EST | PAYCOM1.wsbind

CICS Web Service Binding File (WSBind) Viewer

▼ Maintenance Information

Timestamp: 201402181248

Product: Interpretive XML Conversion

▼ Service Interface and Pipeline Properties

Service mode: Service Provider

Provider URI: /cics/services/PAYCOM1

Requester URI:

WSDL binding name: PAYCOM1HTTPSoapBinding

Operations: PAYCOM1Operation

Transaction ID:

User ID:

Syncpoint: false

▼ Required Runtime and Mapping Levels

Mapping level: 3.0

Runtime level: 3.0

▼ Target Program Interface and Properties

Program name: PAYCOM1

Program interface: COMMAREA

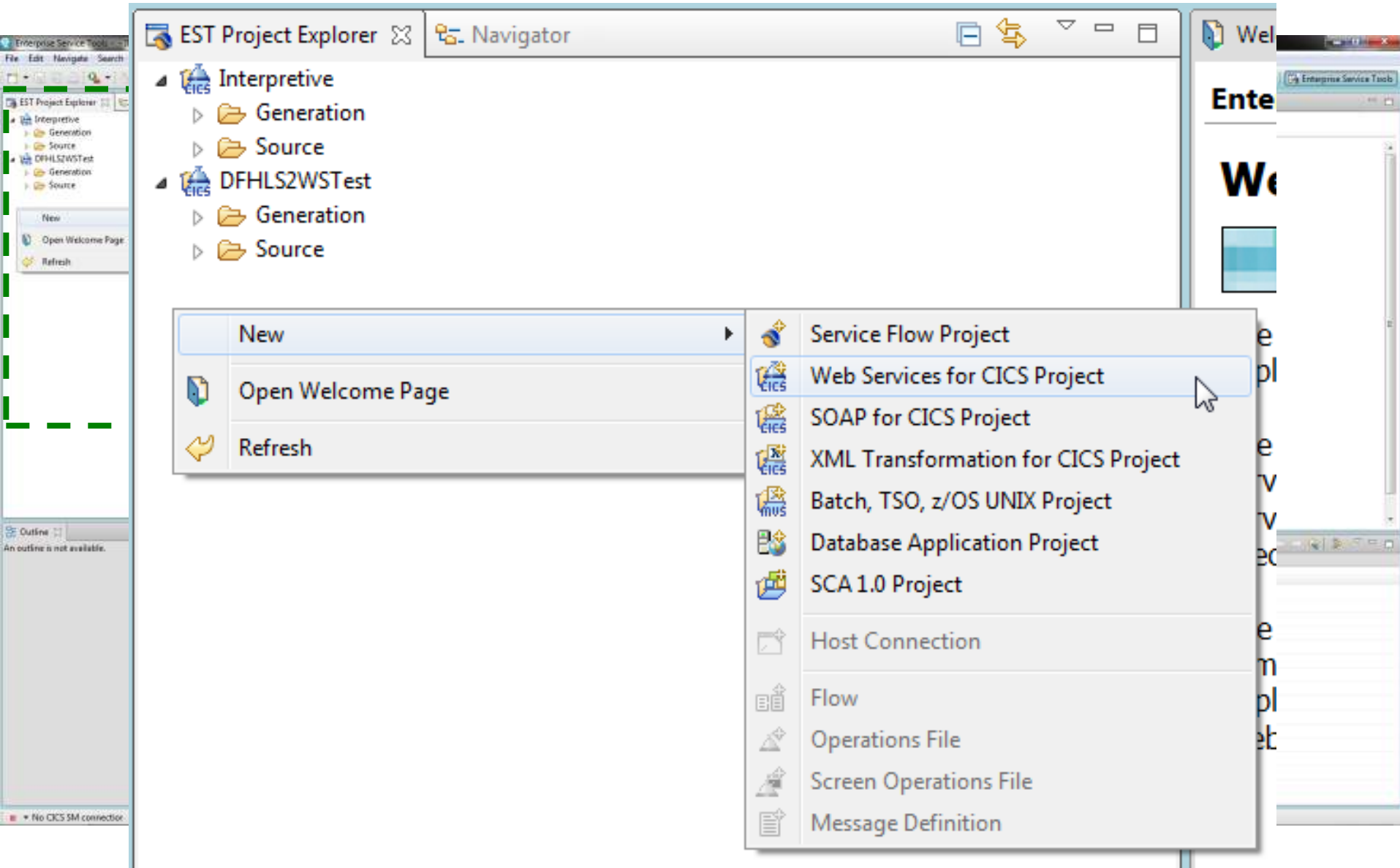
Container name:

Request Channel:

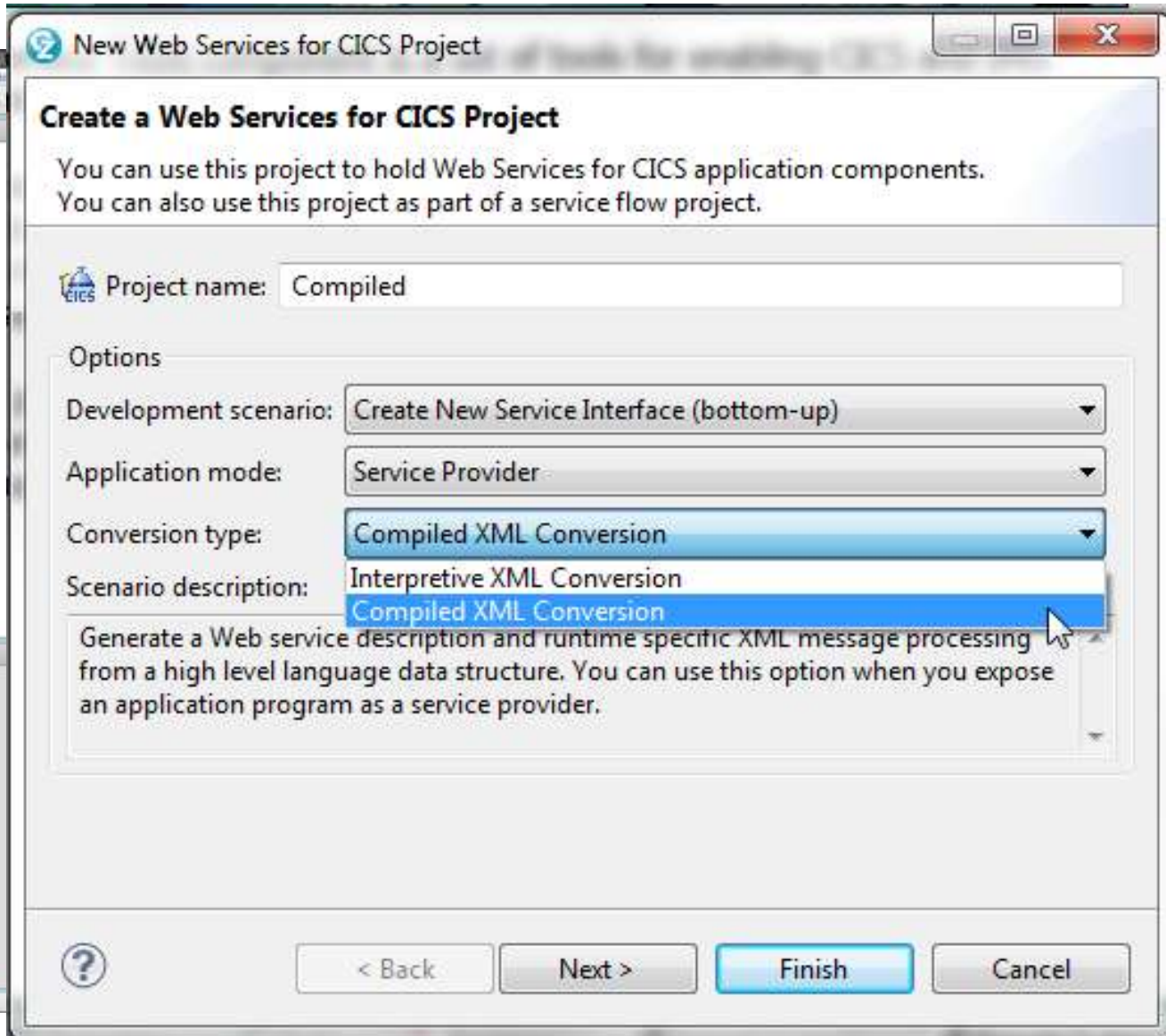
Response Channel:

Vendor Converter name:

Creating a provider using RDz: compiled (1 of 6)



Creating a provider using RDz: compiled (2 of 6)



New Web Services for CICS Project

Create a Web Services for CICS Project

You can use this project to hold Web Services for CICS application components.
You can also use this project as part of a service flow project.

Project name:

Options

Development scenario:

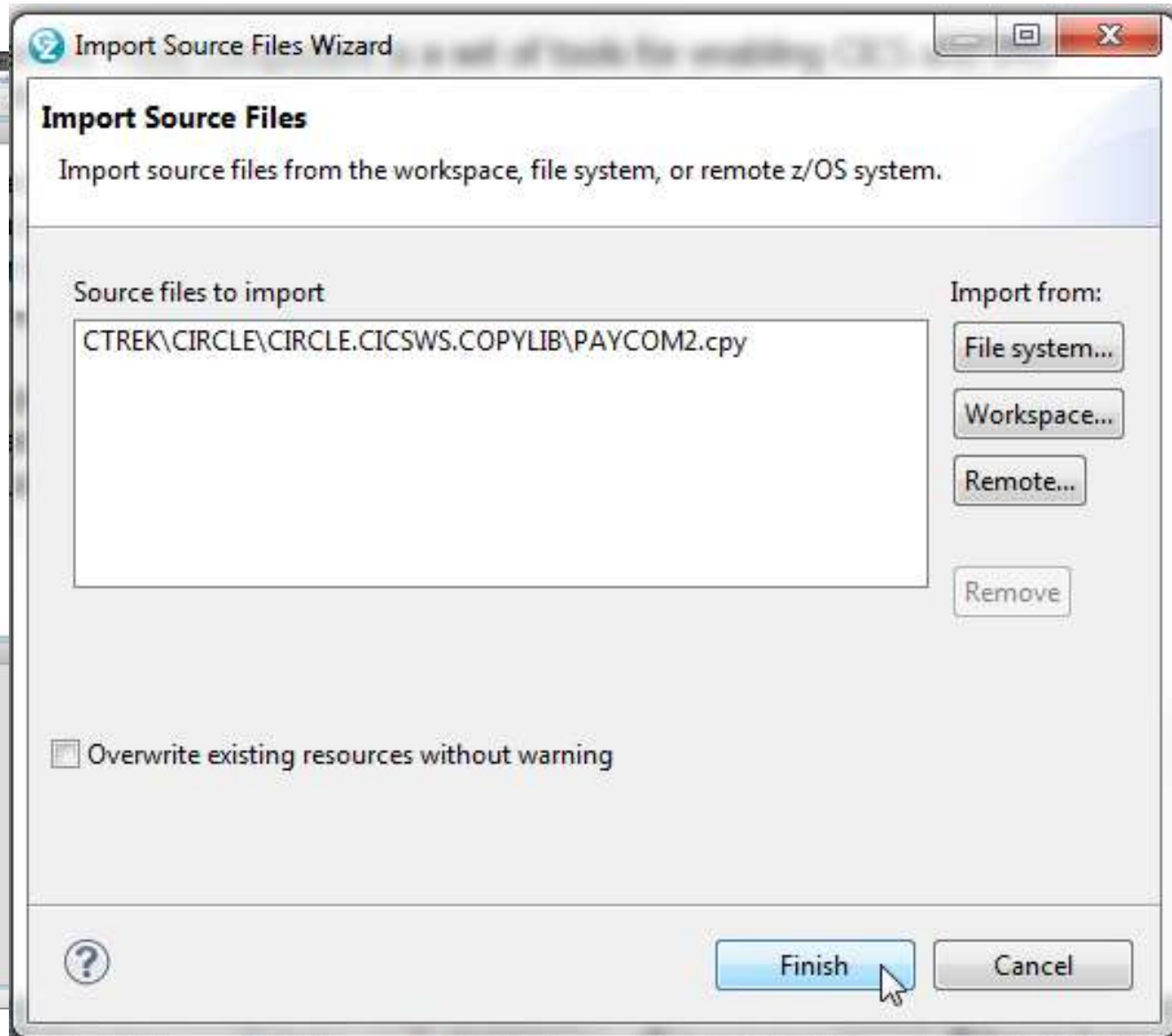
Application mode:

Conversion type:

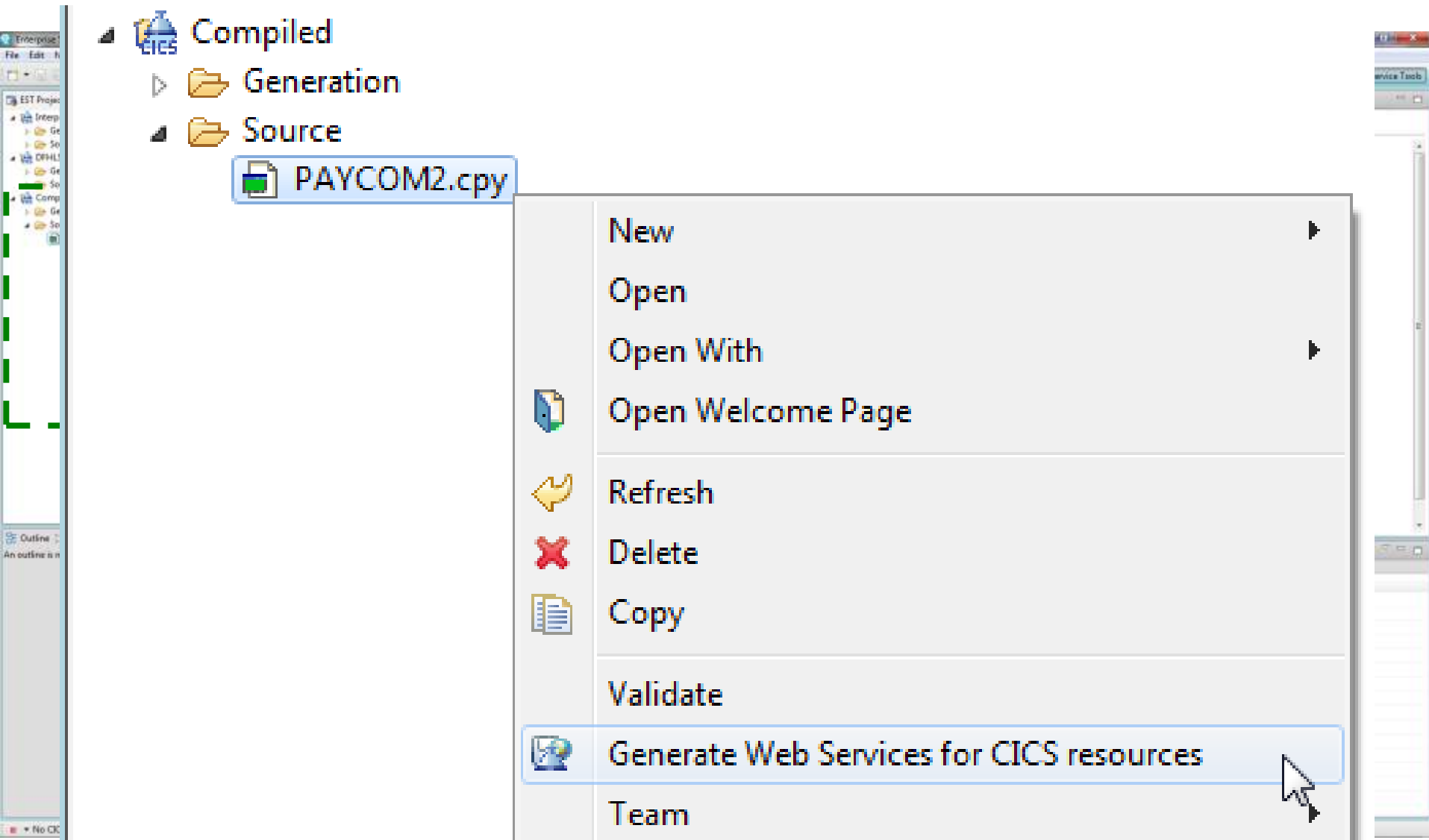
Scenario description:

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

Creating a provider using RDz: compiled (3 of 6)



Creating a provider using RDz: compiled (4 of 6)



The screenshot shows the Enterprise Architect (EES) software interface. On the left, a project tree is visible under the 'Enterprise' window. The tree structure is as follows:

- Enterprise
 - File Edit H
 - EST Project
 - Interp
 - Ge
 - So
 - DPHLI
 - Ge
 - So
 - Comp
 - Ge
 - So

The 'Compiled' folder is expanded, showing the following items:

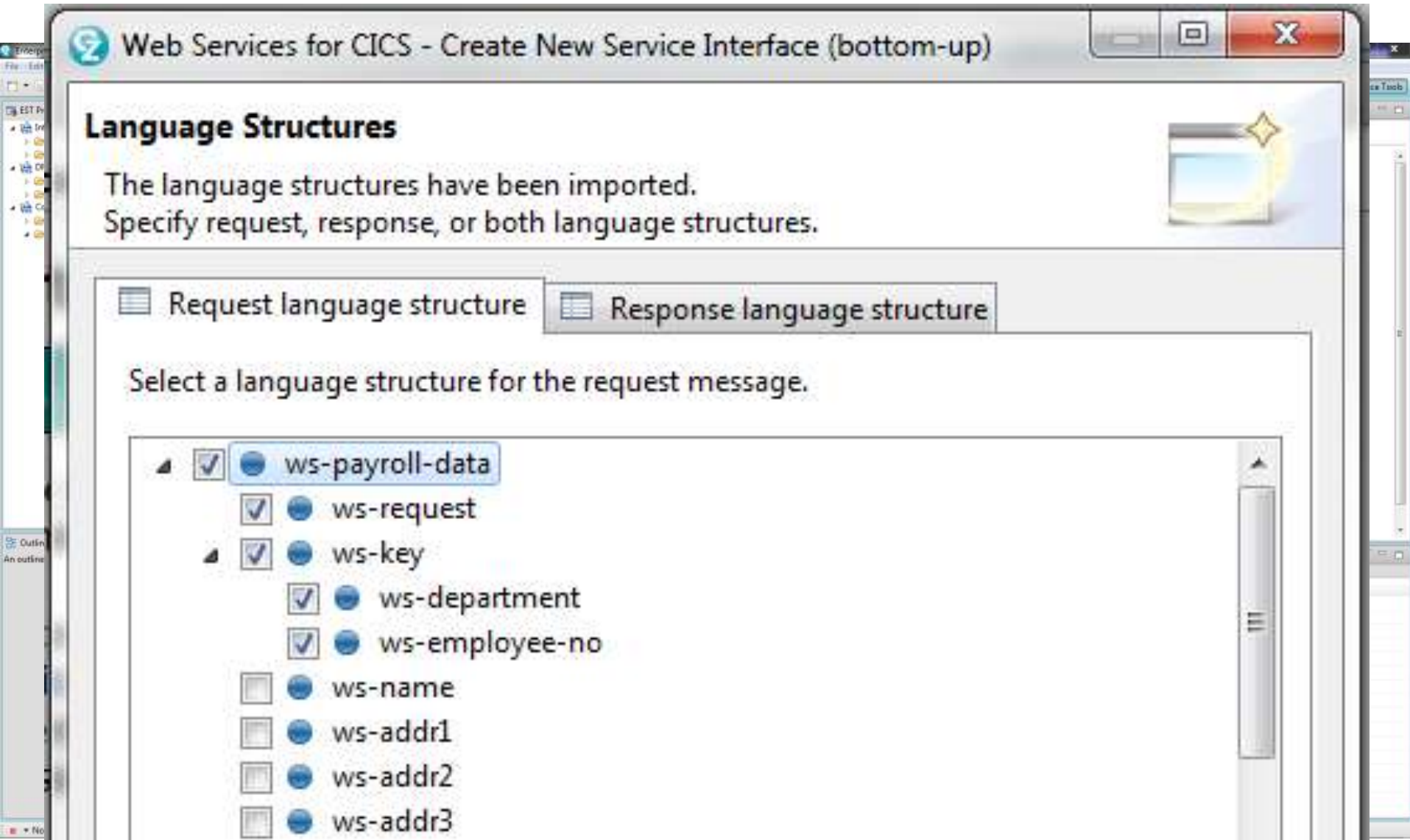
- Generation
- Source
- PAYCOM2.cpy

A right-click context menu is open over the 'PAYCOM2.cpy' file. The menu options are:

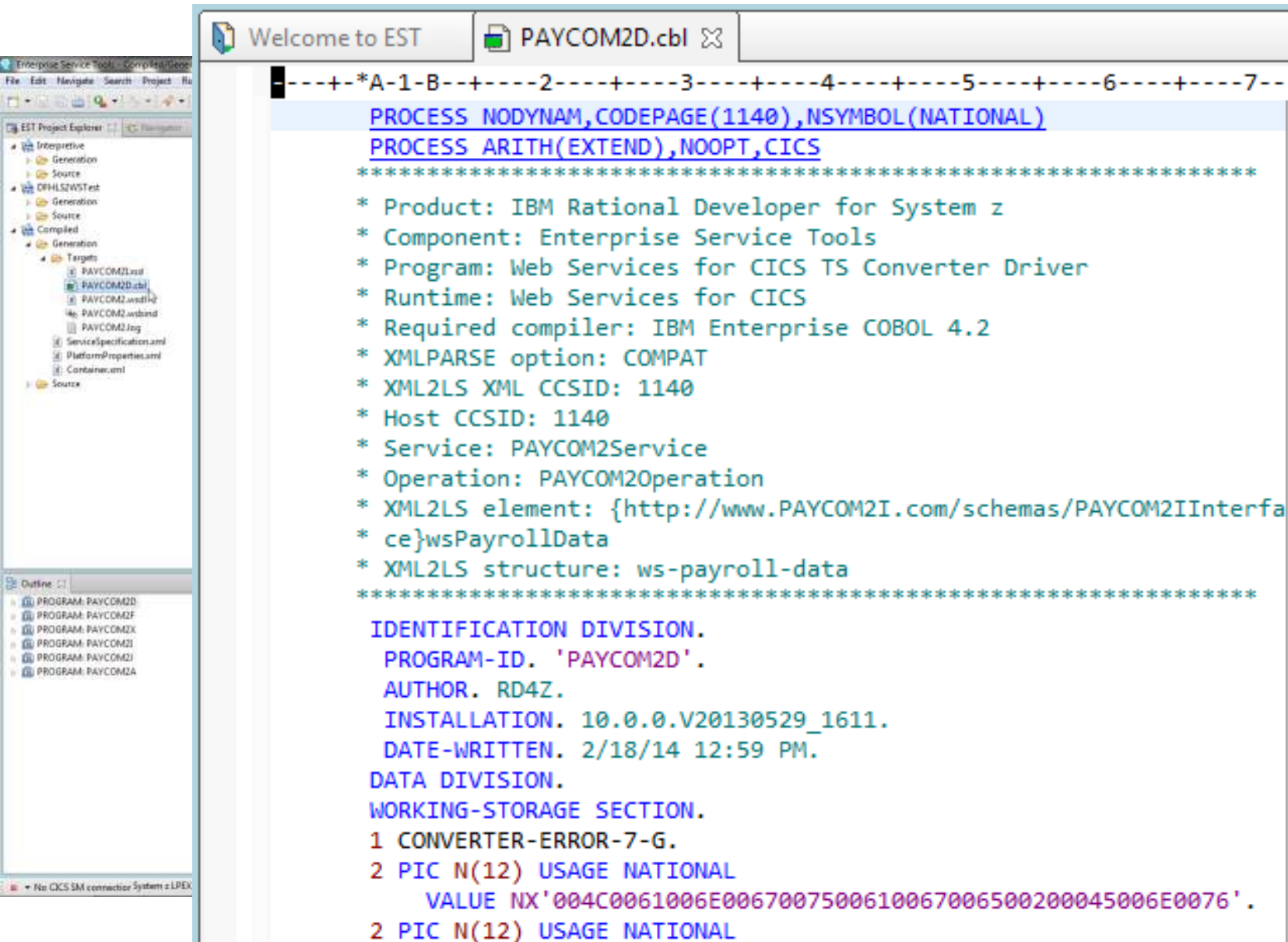
- New
- Open
- Open With
- Open Welcome Page
- Refresh
- Delete
- Copy
- Validate
- Generate Web Services for CICS resources
- Team

The 'Generate Web Services for CICS resources' option is highlighted by the mouse cursor.

Creating a provider using RDz: compiled (5 of 6)



Creating a provider using RDz: compiled (6 of 6)



```
Welcome to EST | PAYCOM2D.cbl
```

```
-----*A-1-B-----2-----3-----4-----5-----6-----7-----|
PROCESS NODYNAM, CODEPAGE(1140), NSYMBOL(NATIONAL)
PROCESS ARITH(EXTEND), NOOPT, CICS
*****
* Product: IBM Rational Developer for System z
* Component: Enterprise Service Tools
* Program: Web Services for CICS TS Converter Driver
* Runtime: Web Services for CICS
* Required compiler: IBM Enterprise COBOL 4.2
* XMLPARSE option: COMPAT
* XML2LS XML CCSID: 1140
* Host CCSID: 1140
* Service: PAYCOM2Service
* Operation: PAYCOM2Operation
* XML2LS element: {http://www.PAYCOM2I.com/schemas/PAYCOM2IInterfa
* ce}wsPayrollData
* XML2LS structure: ws-payroll-data
*****
IDENTIFICATION DIVISION.
    PROGRAM-ID. 'PAYCOM2D'.
    AUTHOR. RD4Z.
    INSTALLATION. 10.0.0.V20130529_1611.
    DATE-WRITTEN. 2/18/14 12:59 PM.
DATA DIVISION.
WORKING-STORAGE SECTION.
1 CONVERTER-ERROR-7-G.
2 PIC N(12) USAGE NATIONAL
    VALUE NX'004C0061006E0067007500610067006500200045006E0076'.
2 PIC N(12) USAGE NATIONAL
```

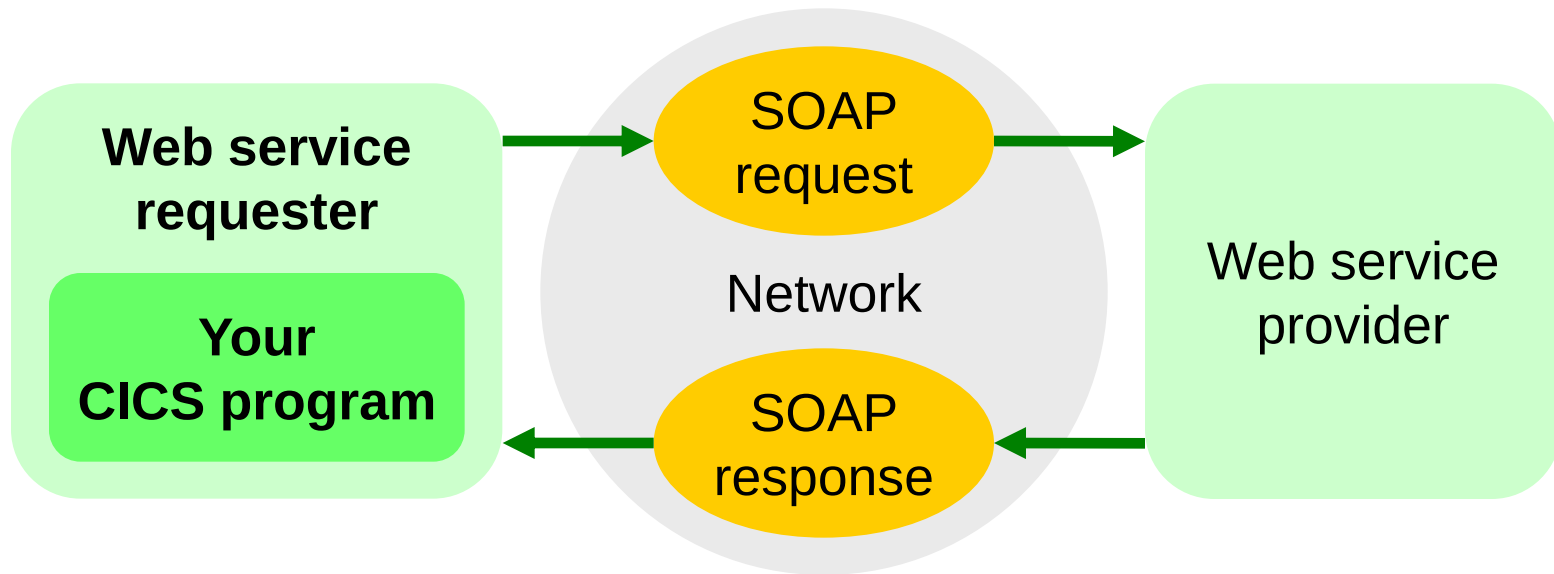
Creating a provider using RDz: after running the RDz wizard

1. Transfer the wsbind file to the z/OS UNIX pickup directory. Optionally, transfer the WSDL file to the same directory.
2. Compiled option only (generated wrapper program):
 - Compile and link the COBOL source program
 - Create a PROGRAM resource
3. Issue a PIPELINE SCAN command.

Creating a provider using RDz Service Flow Modeler

1. In RDz, create a Service Flow Project. This starts a wizard that directs you to:
2. Define a host connection (to the z/OS system mainframe that hosts your CICS application).
3. Navigate to the “start” screen (signon to CICS, start the transaction, clear the screen).
4. Start recording the “flow” (your input, and the transaction output).
5. For each input field (request data), specify a variable name.
6. For each output field (response data), highlight the item on the screen, and specify a variable name.
7. Stop recording. This generates a .seqflow file.
8. Right-click the .seqflow file, and select New Generation Properties File to generate a WSDL file.
9. Click Generate Runtime code. (This wizard can submit the compile JCL on z/OS for you.)
10. The generated code includes a web service provider COBOL program that drives your original CICS application.

Creating a web service requester in CICS

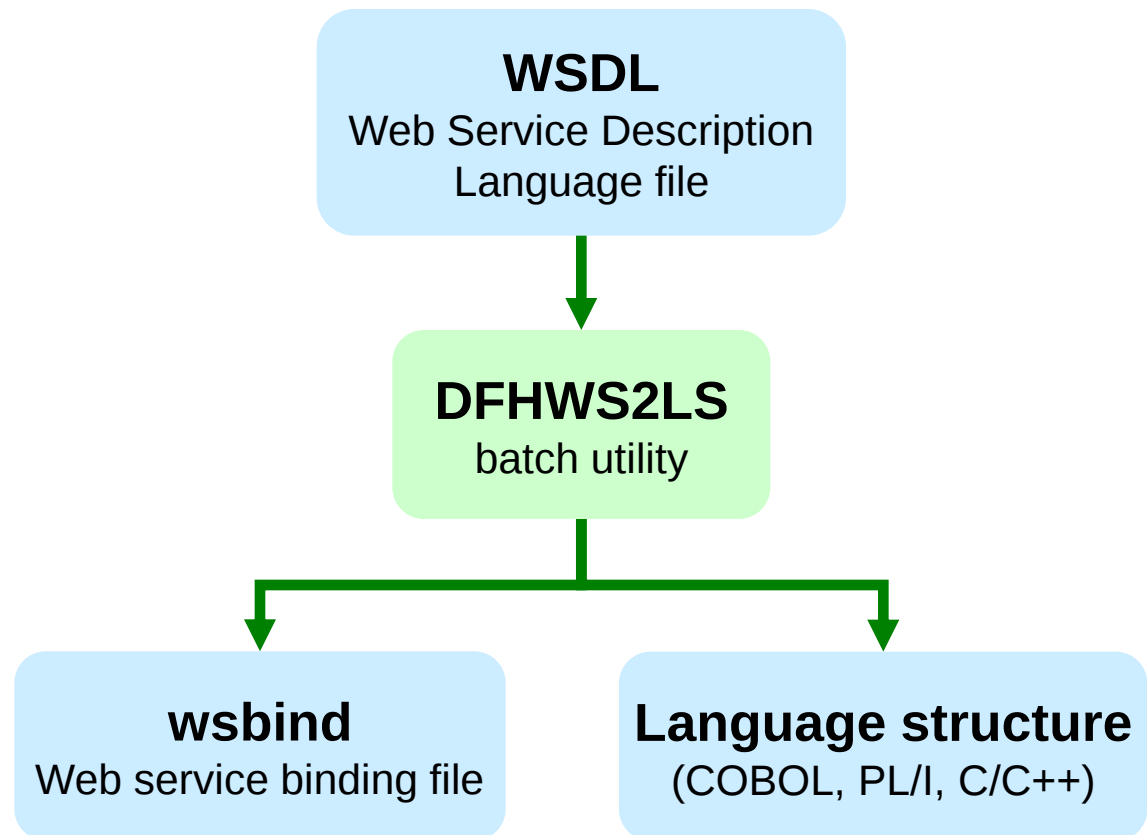


Methods for creating a web service requester in CICS

- 1. CICS web services assistant** from a WSDL, using the DFHWS2LS batch utility
 - 2. RDz** from a WSDL (using a wizard), with interpretive runtime XML conversion, as per DFHWS2LS, above (no compiled option for a requester)
- Both methods generate copybooks and a wsbind file. However, the RDz also generates COBOL source for a requester program, demonstrating how to use the EXEC CICS INVOKE WEBSERVICE command.

Creating a requester using the CICS web services assistant

- **You will need:** the WSDL for the web service that you want to use



Creating the CICS infrastructure for a requester

- Identical to the steps for a provider, except that a requester does not require a TCPIP SERVICE or a URIMAP resource
1. Create a **pipeline configuration file**.
 2. Create a **PIPELINE** resource.
 3. Unless you use autoinstalled PROGRAM definitions, create a **PROGRAM** resource for each program in the pipeline.

Creating a requester using the CICS web services assistant

1. Run the **DFHWS2LS** batch utility (for example, specifying a COBOL copybook as the input file).
2. Copy the generated **wsbind** file to the pickup directory (the z/OS UNIX path specified by the WSDIR attribute of the PIPELINE resource).
Optionally, copy the generated **WSDL** file to the same path.
3. Install the **PIPELINE** (dynamically creates the WEBSERVICE resource).
4. Add an **EXEC CICS INVOKE WEBSERVICE** command to your COBOL program to send the request, and additional code to process the response.

The requester is ready for testing.

JCL to run DFHWS2LS

```
//SYSEGXLS JOB (39248C,A,T),'LS2WS',  
// MSGCLASS=A,NOTIFY=&SYSUID,REGION=0M  
// SET QT=''''  
//WHERE SMA JCLLIB ORDER=CIRCLE.CICSWS.PROCLIB  
//JAVAPROG EXEC DFHWS2LS,  
// JAVADIR='Java601_64/J6.0.1_64',PATHPREF='/u',TMPDIR='/u/tmp',  
// TMPFILE=&QT.&SYSUID.&QT,USSDIR='cicsts42'  
//INPUT.SYSUT1 DD *  
PDSLIB=CIRCLE.CICSWS.COPYLIB  
REQMEM=REQCOM  
RESPMEM=RESCOM  
MAPPING-LEVEL=3.0  
MINIMUM-RUNTIME-LEVEL=CURRENT  
LANG=COBOL  
WSBIND=/u/usr/lpp/cicsts/cicsts42/samples/webservices/wsbind/requester/*  
paybus6.wsbind  
WSDL=/u/usr/lpp/cicsts/cicsts42/samples/webservices/wsd1/paybus.wsdl  
LOGFILE=/u/sysegx0/paybus6  
/*
```

Output COBOL copybook PDS members:
one for the request, another for the
response

Output wsbind file

Input WSDL file

COBOL copybook generated by DFHWS2LS

```

03 PAYBUSOperation.
  06 wsXpayrollXdata.
    09 wsXrequest      PIC X(4).
    09 wsXkey.
      12 wsXdepartment  PIC X(1).
      12 wsXemployeeXno PIC X(5).
    09 wsXname          PIC X(20).
    09 wsXaddr1          PIC X(20).
    09 wsXaddr2          PIC X(20).
    09 wsXaddr3          PIC X(20).
    09 wsXphoneXno       PIC X(8).
    09 wsXtimestamp      PIC X(8).
    09 wsXsalary         PIC X(8).
    09 wsXstartXdate     PIC X(8).
    09 wsXremarks        PIC X(32).
    09 wsXmsg            PIC X(60).
    ...
  
```

Corresponding XML snippet

```

<wsXpayrollXdata>
  <wsXrequest>DISP</wsXrequest>
  <wsXkey>
    <wsXdepartment>1</wsXdepartment>
    <wsXemployeeXno>00001</wsXemployeeXno>
  </wsXkey>
  <wsXname>CIRCLE COMPUTER 1 </wsXname>
  ...

```

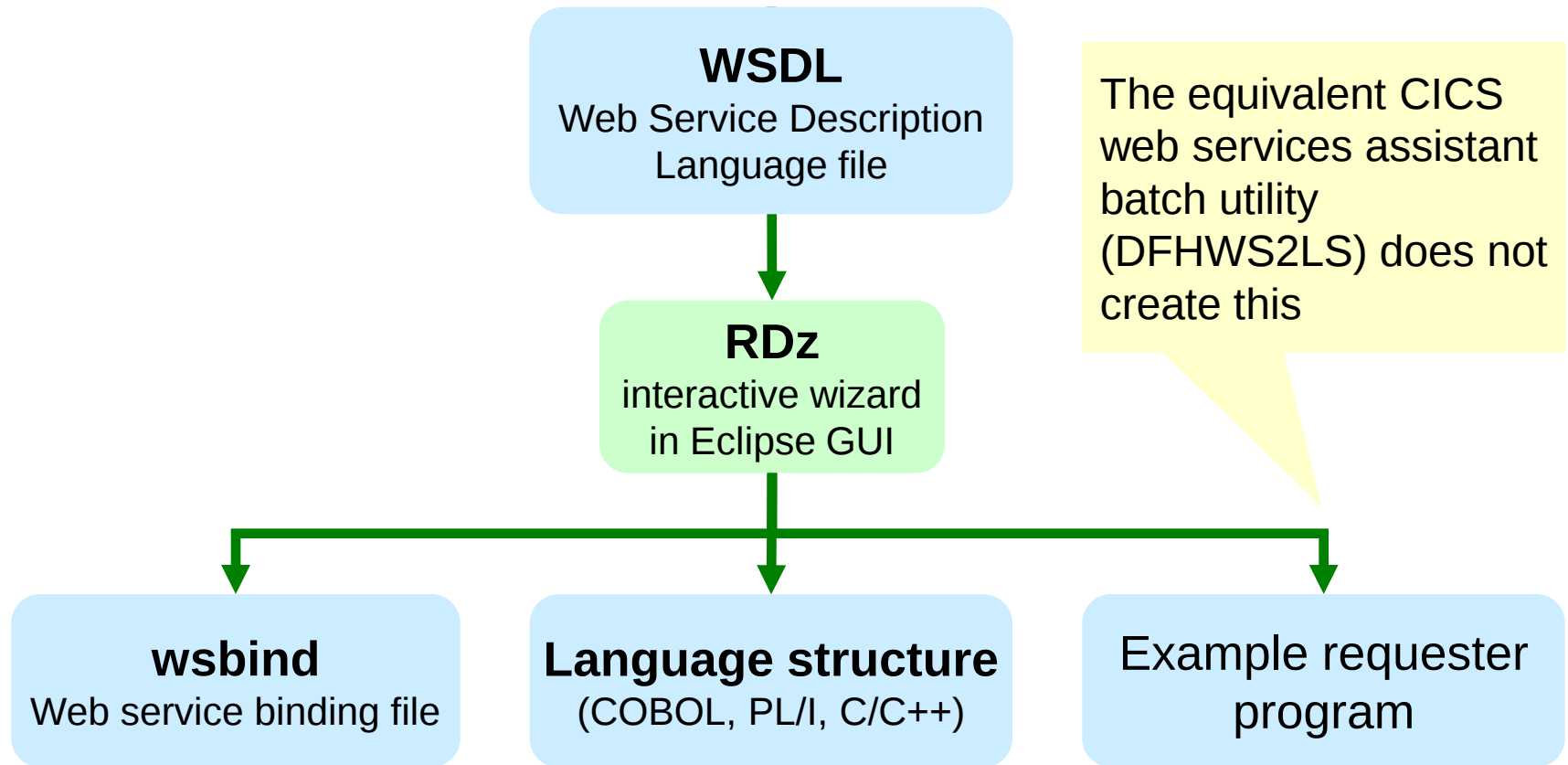
XML allows hyphens in element names, but some applications and programming languages interpret such hyphens as minus signs (mathematical operators), with undesirable results

Sending a request to a web service from a CICS COBOL program

```
EXEC CICS INVOKE  
  WEBSERVICE(CV-WEBSERVICE)  
  CHANNEL(CV-CHANNEL-NAME)  
  OPERATION(CV-OPERATION)  
  URI(CV-URI)  
  RESP(WS-EIB-RESP)  
END-EXEC.
```

The RDz wizard generates a sample CICS COBOL program that does this

Creating a requester using RDz



Creating a requester using RDz (1 of 8)



The screenshot shows the 'EST Project Explorer' and 'Navigator' windows. The 'EST Project Explorer' window displays a tree view of projects and their subfolders. The 'Navigator' window is currently empty. A context menu is open over the 'EST Project Explorer' window, showing options for creating new projects.

EST Project Explorer

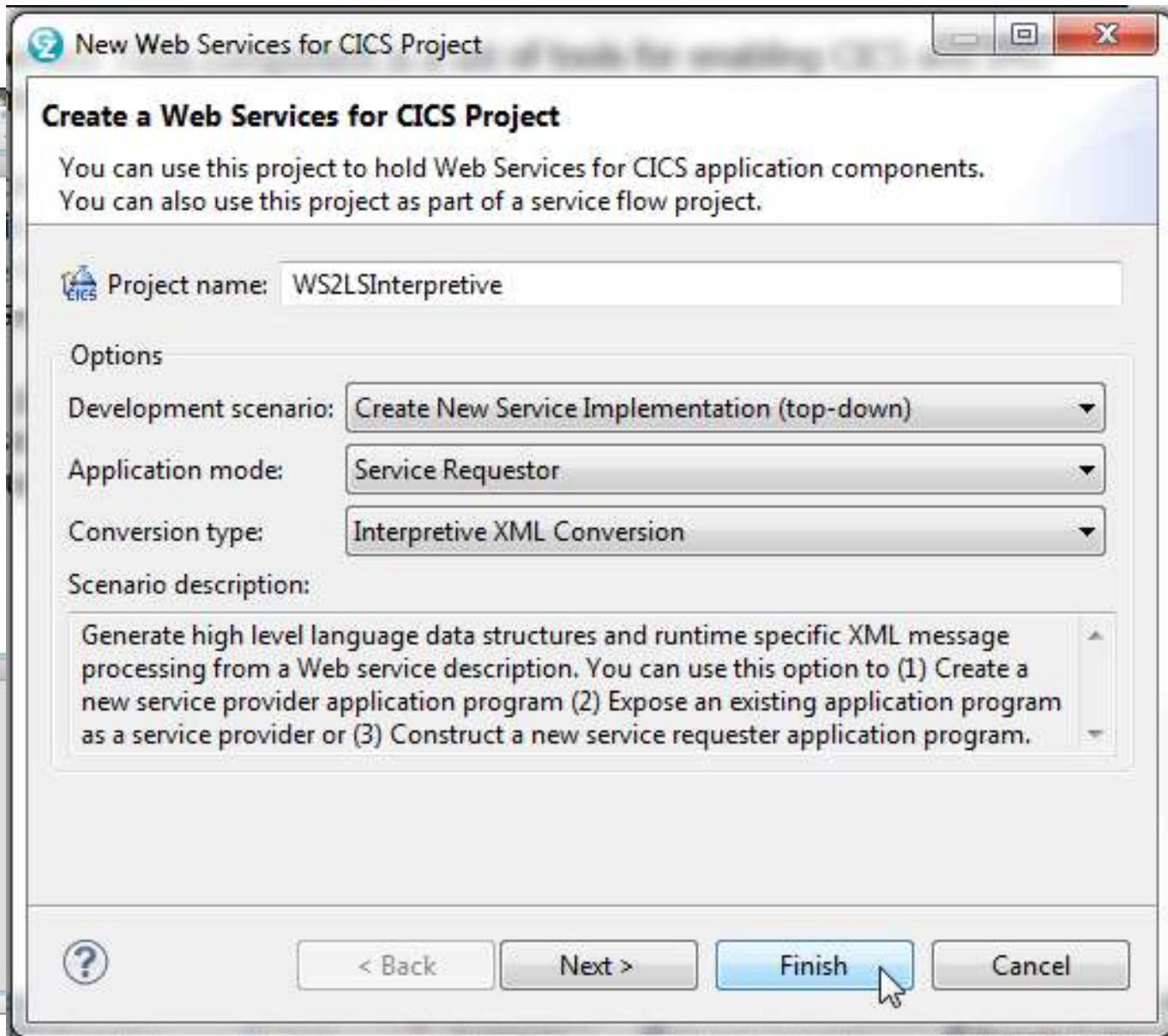
- Interpretive
 - Generation
 - Source
- DFHLS2WSTest
 - Generation
 - Source
- Compiled
 - Generation
 - Source

Navigator

Context Menu:

- New
 - Service Flow Project
 - Web Services for CICS Project
 - SOAP for CICS Project
 - XML Transformation for CICS Project
 - Batch, TSO, z/OS UNIX Project
 - Database Application Project
 - SCA 1.0 Project
- Open Welcome Page
- Refresh

Creating a requester using RDz (2 of 8)



New Web Services for CICS Project

Create a Web Services for CICS Project

You can use this project to hold Web Services for CICS application components.
You can also use this project as part of a service flow project.

Project name: WS2LSInterpretive

Options

Development scenario: Create New Service Implementation (top-down)

Application mode: Service Requestor

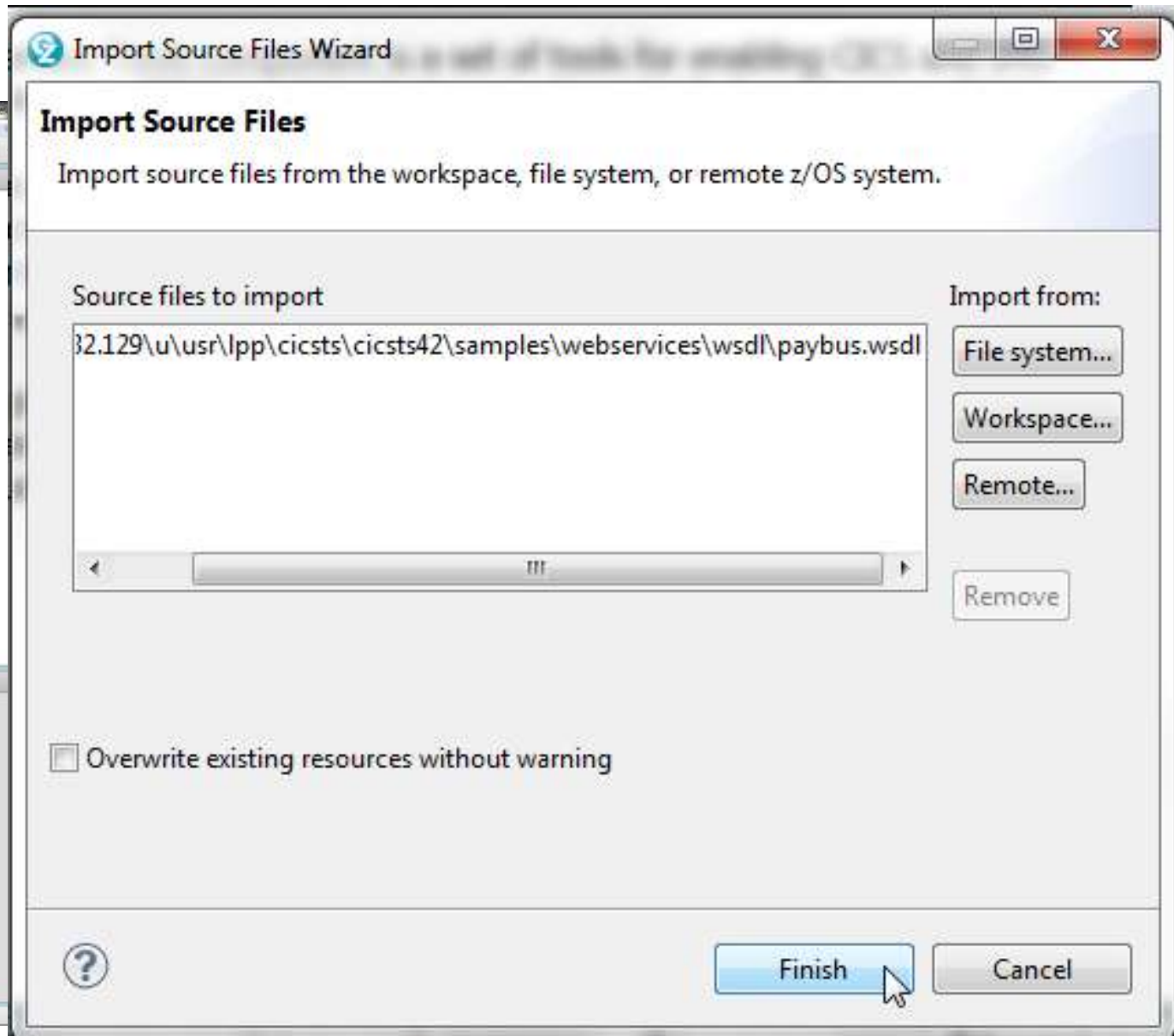
Conversion type: Interpretive XML Conversion

Scenario description:

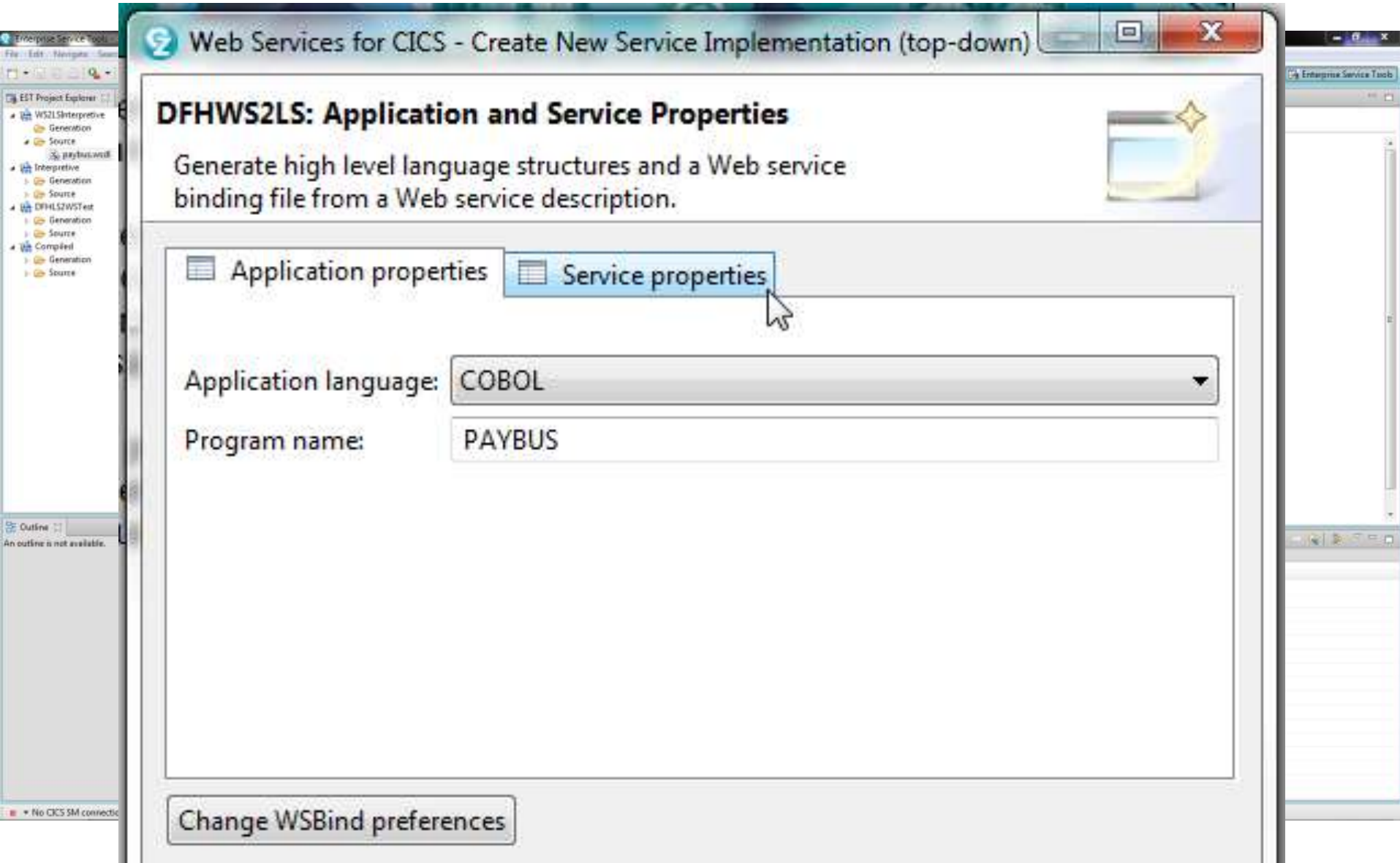
Generate high level language data structures and runtime specific XML message processing from a Web service description. You can use this option to (1) Create a new service provider application program (2) Expose an existing application program as a service provider or (3) Construct a new service requester application program.

Navigation: ? < Back Next > **Finish** Cancel

Creating a requester using RDz (3 of 8)



Creating a requester using RDz (4 of 8)



The screenshot shows the 'Web Services for CICS - Create New Service Implementation (top-down)' dialog box. The title bar indicates the application is 'Enterprise Service Tools'. The main window has a tabbed interface with two tabs: 'Application properties' and 'Service properties'. The 'Service properties' tab is selected and highlighted with a mouse cursor. Below the tabs, there are two input fields: 'Application language:' with a dropdown menu showing 'COBOL', and 'Program name:' with a text box containing 'PAYBUS'. At the bottom of the dialog, there is a button labeled 'Change WSBind preferences'. The background shows the 'EST Project Explorer' on the left with a tree view containing folders like 'WS2LSInterpretive', 'DFHWS2LS', and 'DFHWS2LSTest', each with sub-items for 'Generation', 'Source', and 'Compiled'. A status bar at the bottom left indicates 'No CICS SM connection'.

Web Services for CICS - Create New Service Implementation (top-down)

DFHWS2LS: Application and Service Properties

Generate high level language structures and a Web service binding file from a Web service description.

☐ Application properties ☒ Service properties

Application language: COBOL

Program name: PAYBUS

Change WSBind preferences

Creating a requester using RDz (5 of 8)

Web Services for CICS - Create New Service Implementation (top-down)

DFHWS2LS: Application and Service Properties

Generate high level language structures and a Web service binding file from a Web service description.

Application properties | Service properties

Local URI: /cics/services/paybus

WSDL service:

Binding element: PAYBUSHTTPSoapBinding

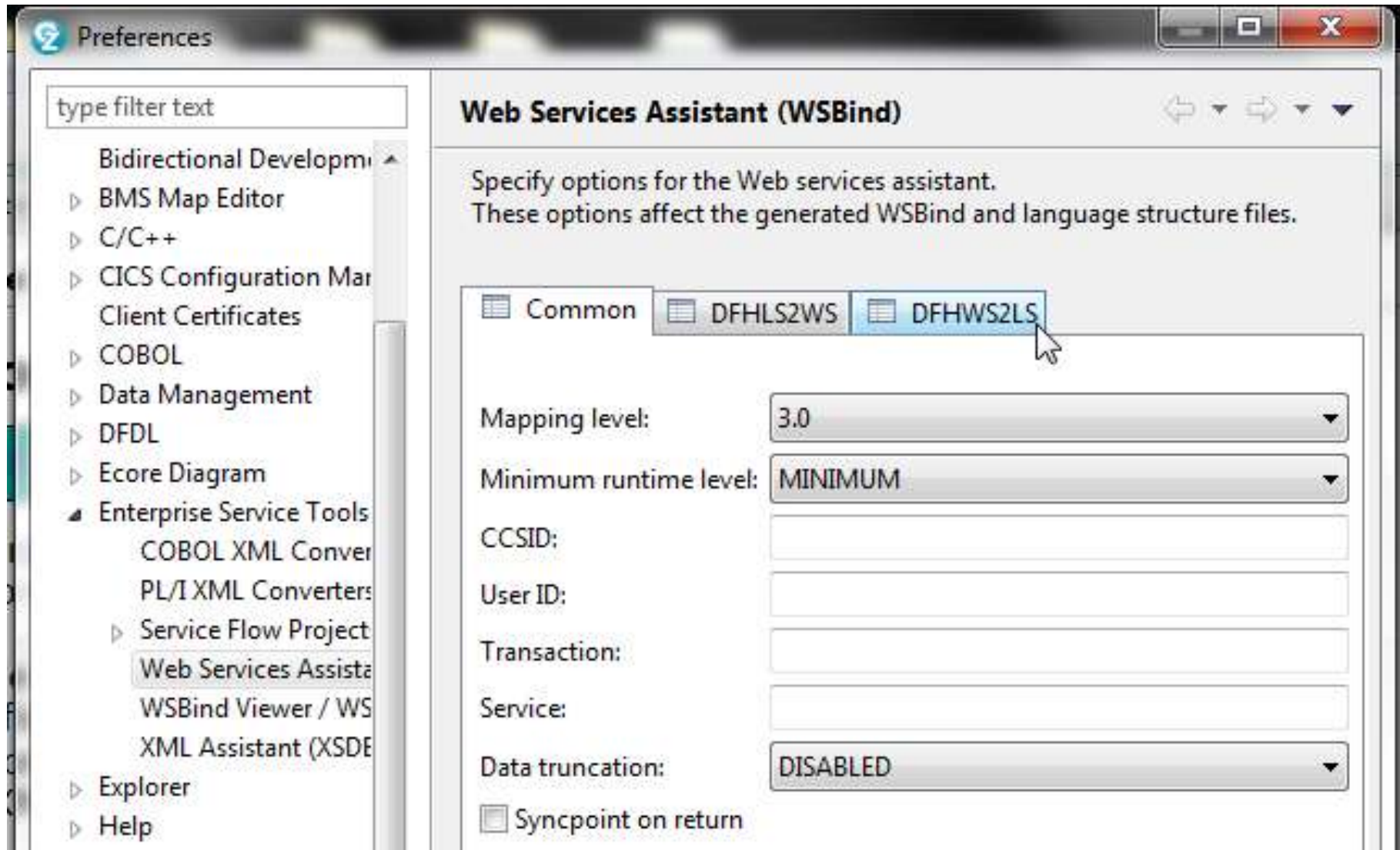
Available operations:

☒ PAYBUSOperation

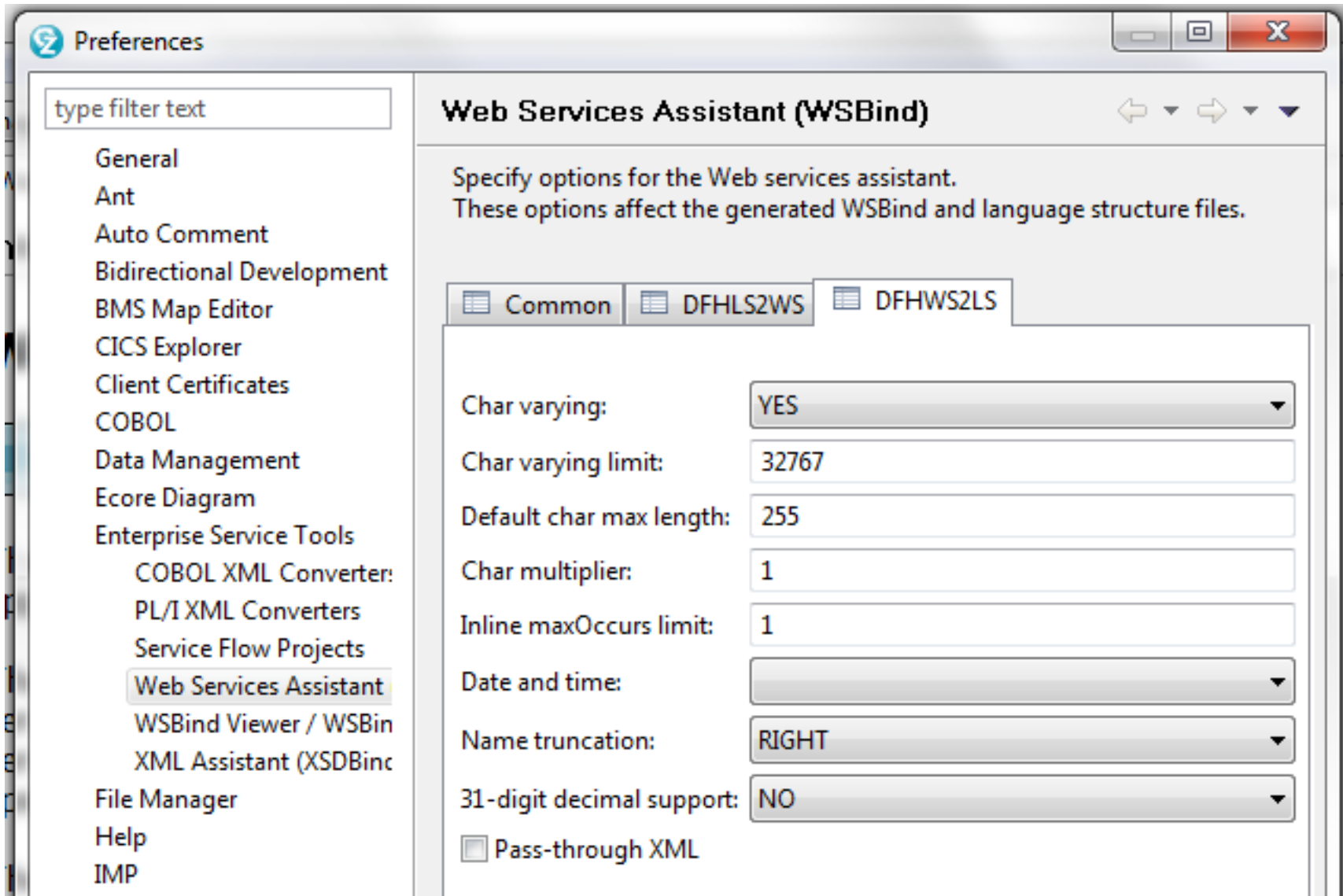
Select all Deselect all

Change WSBind preferences

Creating a requester using RDz (6 of 8)



Creating a requester using RDz (7 of 8)



The image shows a screenshot of the 'Web Services Assistant (WSBind)' preferences dialog box. The dialog has a title bar with a minimize, maximize, and close button. On the left is a sidebar with a search filter 'type filter text' and a list of categories. The 'Web Services Assistant' category is selected. The main area is titled 'Web Services Assistant (WSBind)' and contains instructions: 'Specify options for the Web services assistant. These options affect the generated WSBind and language structure files.' Below this are three tabs: 'Common', 'DFHLS2WS', and 'DFHWS2LS'. The 'Common' tab is active, showing several settings: 'Char varying' (YES), 'Char varying limit' (32767), 'Default char max length' (255), 'Char multiplier' (1), 'Inline maxOccurs limit' (1), 'Date and time' (empty dropdown), 'Name truncation' (RIGHT), '31-digit decimal support' (NO), and an unchecked 'Pass-through XML' checkbox.

Preferences

type filter text

- General
- Ant
- Auto Comment
- Bidirectional Development
- BMS Map Editor
- CICS Explorer
- Client Certificates
- COBOL
- Data Management
- Ecore Diagram
- Enterprise Service Tools
 - COBOL XML Converter:
 - PL/I XML Converters
 - Service Flow Projects
 - Web Services Assistant**
 - WSBind Viewer / WSBin
 - XML Assistant (XSDBinc)
- File Manager
- Help
- IMP

Web Services Assistant (WSBind)

Specify options for the Web services assistant.
These options affect the generated WSBind and language structure files.

Common DFHLS2WS DFHWS2LS

Char varying: YES

Char varying limit: 32767

Default char max length: 255

Char multiplier: 1

Inline maxOccurs limit: 1

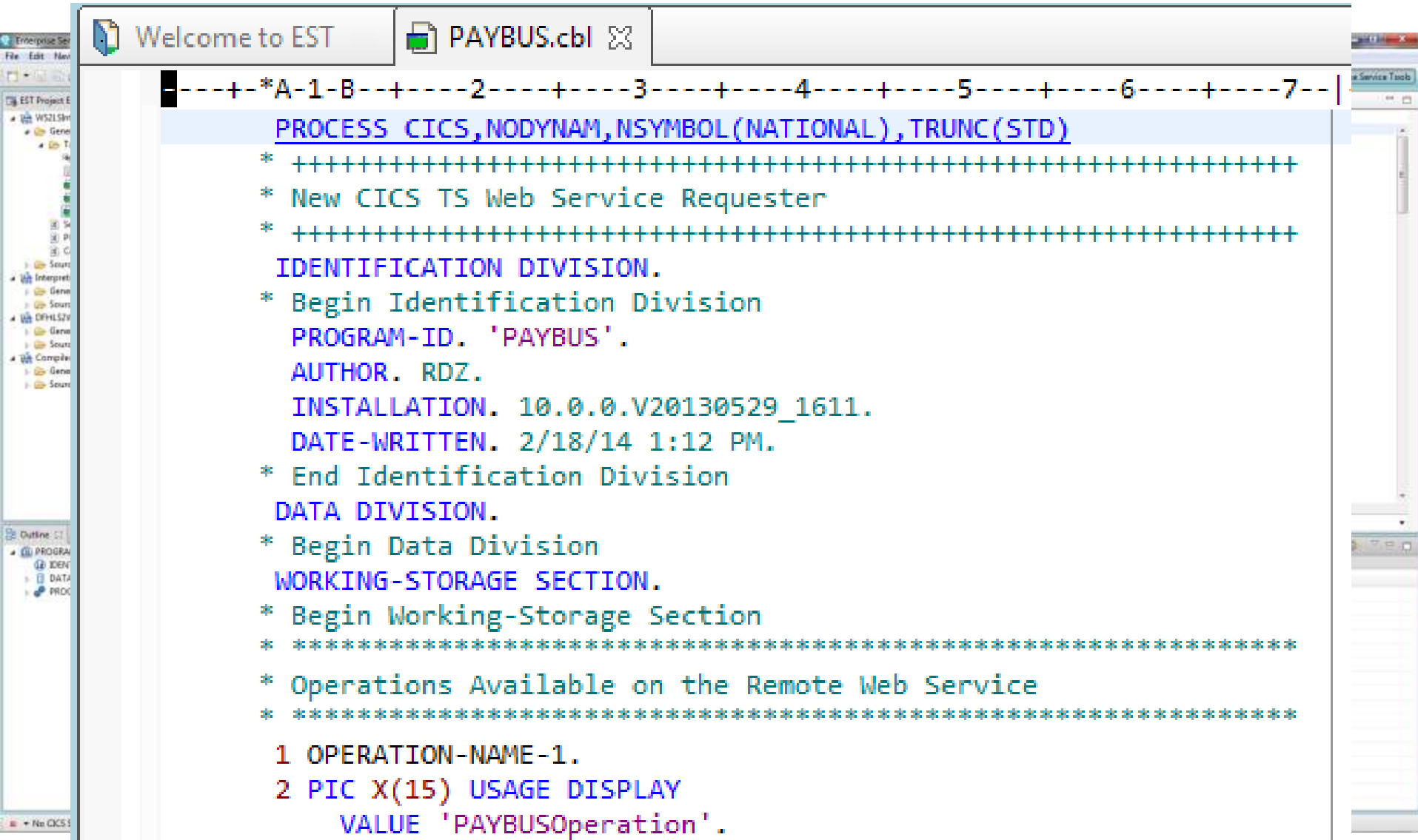
Date and time:

Name truncation: RIGHT

31-digit decimal support: NO

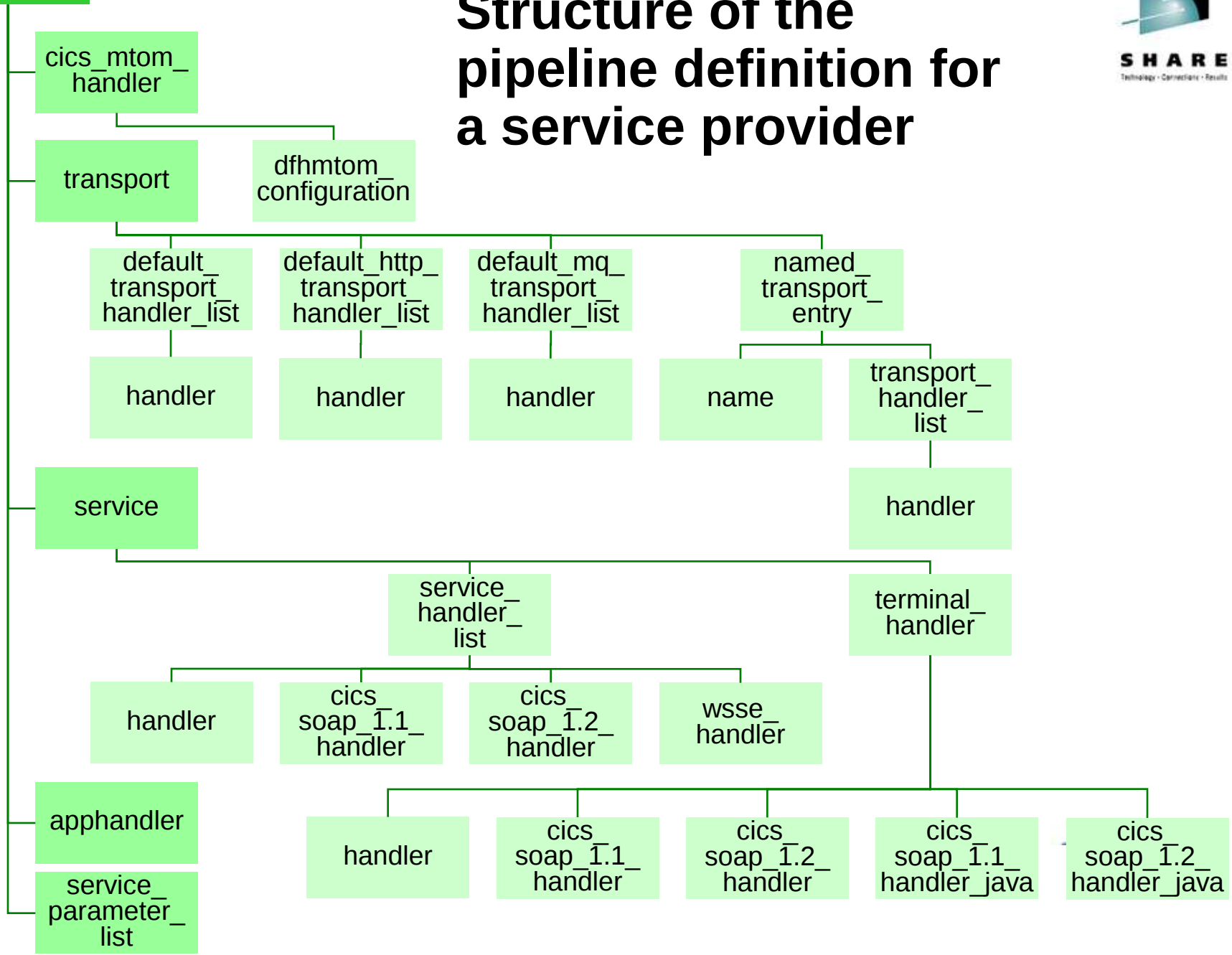
☐ Pass-through XML

Creating a requester using RDz (8 of 8)



```
--+--*A-1-B--+--2--+--3--+--4--+--5--+--6--+--7--+  
PROCESS CICS,NODYNAM,NSYMBOL(NATIONAL),TRUNC(STD)  
* ++++++  
* New CICS TS Web Service Requester  
* ++++++  
IDENTIFICATION DIVISION.  
* Begin Identification Division  
PROGRAM-ID. 'PAYBUS'.  
AUTHOR. RDZ.  
INSTALLATION. 10.0.0.V20130529_1611.  
DATE-WRITTEN. 2/18/14 1:12 PM.  
* End Identification Division  
DATA DIVISION.  
* Begin Data Division  
WORKING-STORAGE SECTION.  
* Begin Working-Storage Section  
* *****  
* Operations Available on the Remote Web Service  
* *****  
1 OPERATION-NAME-1.  
2 PIC X(15) USAGE DISPLAY  
VALUE 'PAYBUSOperation'.
```

Structure of the pipeline definition for a service provider



Diagnosing web services in CICS: sniffing containers in the pipeline

- The IBM Redbook *Implementing CICS Web Services*, SG24-7206, presents a simple “sniffer” program that displays (in tdqueue CESE) the contents of the containers available in the pipeline.
- To use the sniffer, you add it to the pipeline (configuration file) as a message handler.

- For example, in a provider pipeline:

```
<provider_pipeline>  
  <service>  
    <service_handler_list>  
      <handler>  
        <program>SNIFFER</program>  
        <handler_parameter_list/>  
      </handler>  
    </service_handler_list>  
    <terminal_handler>  
      <cics_soap_1.1_handler/>  
    </terminal_handler>  
  </service>  
  <apphandler>DFHPITP</apphandler>  
</provider_pipeline>
```

Sniffer output (1 of 5)

```

CPIH 20120314113934 SNIFFER : *** Start ***
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHFUNCTION
CPIH 20120314113934 SNIFFER : Content length     : 00000016
CPIH 20120314113934 SNIFFER : Container content: RECEIVE-REQUEST
CPIH 20120314113934 SNIFFER : Containers on channel: List starts.
CPIH 20120314113934 SNIFFER : >=====<
...
CPIH 20120314113934 SNIFFER : Container Name      : DFHFUNCTION
CPIH 20120314113934 SNIFFER : Content length     : 00000016
CPIH 20120314113934 SNIFFER : Container content: RECEIVE-REQUEST
CPIH 20120314113934 SNIFFER : >=====<
...
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-URI
CPIH 20120314113934 SNIFFER : Content length     : 00000008
CPIH 20120314113934 SNIFFER : Container content: /paybus1
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHREQUEST
CPIH 20120314113934 SNIFFER : Content length     : 00002928
CPIH 20120314113934 SNIFFER : Container content:
<SOAP-ENV:Envelope ... >
  <SOAP-ENV:Body ... >
    <PAYBUSOperationRequest>
      <ws_payroll_data>
        <ws_request>DISP</ws_request>
        <ws_key>
          <ws_department>1</ws_department>
          <ws_employee_no>00001</ws_employee_no>
        </ws_key>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
  
```

Sniffer output (2 of 5)



```
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-PIPELINE
CPIH 20120314113934 SNIFFER : Content length   : 00000008
CPIH 20120314113934 SNIFFER : Container content: CICSWSS
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-USERID
CPIH 20120314113934 SNIFFER : Content length     : 00000008
CPIH 20120314113934 SNIFFER : Container content: CICSTS41
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-TRANID
CPIH 20120314113934 SNIFFER : Content length     : 00000004
CPIH 20120314113934 SNIFFER : Container content: CPIH
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-WEBSERVICE
CPIH 20120314113934 SNIFFER : Content length     : 00000032
CPIH 20120314113934 SNIFFER : Container content: paybus1
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-APPHANDLER
CPIH 20120314113934 SNIFFER : Content length     : 00000008
CPIH 20120314113934 SNIFFER : Container content: DFHPITP
CPIH 20120314113934 SNIFFER : Containers on channel: List ends
CPIH 20120314113934 SNIFFER : DFHRESPONSE        container deleted
CPIH 20120314113934 SNIFFER : **** End ****
```



Sniffer output (3 of 5)



```
CPIH 20120314113934 SNIFFER : *** Start ***
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHFUNTION
CPIH 20120314113934 SNIFFER : Content length    : 00000016
CPIH 20120314113934 SNIFFER : Container content: SEND-RESPONSE
CPIH 20120314113934 SNIFFER : Containers on channel: List starts.
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-OUTACTION
CPIH 20120314113934 SNIFFER : Content length      : 00000067
CPIH 20120314113934 SNIFFER : Container content:
C"http://www.PAYBUS.PAYCOM1.com/PAYBUSPort/PAYBUSOperationResponse"
CPIH 20120314113934 SNIFFER : >=====<
...
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-WSDL-CTX
CPIH 20120314113934 SNIFFER : Content length      : 00000116
CPIH 20120314113934 SNIFFER : Container content:
http://www.PAYBUS.PAYCOM1.com PAYBUSOperation
http://www.PAYBUS.PAYCOM1.com
http://www.PAYBUS.PAYCOM1.com PAYBUSPort
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-OPERATION
CPIH 20120314113934 SNIFFER : Content length      : 00000015
CPIH 20120314113934 SNIFFER : Container content: PAYBUSOperation
```

Sniffer output (4 of 5)



```
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHRESPONSE
CPIH 20120314113934 SNIFFER : Content length   : 00002446
CPIH 20120314113934 SNIFFER : Container content:
```

```
<SOAP-ENV:Envelope ... >
  <SOAP-ENV:Body>
    <PAYBUSOperationResponse ... >
      <ws_payroll_data>
        <ws_request>DISP</ws_request>
        <ws_key>
          <ws_department>1</ws_department>
          <ws_employee_no>00001</ws_employee_no>
        </ws_key>
        <ws_name>SHARE</ws_name>
        <ws_addr1>QUEENSBURY HSE</ws_addr1>
        <ws_addr2>BRIGHTON</ws_addr2>
        <ws_addr3>SUSSEX</ws_addr3>
        <ws_phone_no>75529900</ws_phone_no>
        <ws_timestamp></ws_timestamp>
        <ws_salary>1234.56</ws_salary>
        <ws_start_date>28101984</ws_start_date>
        <ws_remarks>CIRCLE IS MAGIC</ws_remarks>
        <ws_msg></ws_msg>
        <ws_upd_inds>
          <ws_upd_name></ws_upd_name>
```

...

Sniffer output (5 of 5)

```
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHFUNCTION
CPIH 20120314113934 SNIFFER : Content length    : 00000016
CPIH 20120314113934 SNIFFER : Container content: SEND-RESPONSE
```

....

```
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-WEBSERVICE
CPIH 20120314113934 SNIFFER : Content length      : 00000032
CPIH 20120314113934 SNIFFER : Container content: paybus1
CPIH 20120314113934 SNIFFER : >=====<
CPIH 20120314113934 SNIFFER : Container Name      : DFHWS-APPHANDLER
CPIH 20120314113934 SNIFFER : Content length      : 00000008
CPIH 20120314113934 SNIFFER : Container content: DFHPITP
CPIH 20120314113934 SNIFFER : Containers on channel: List ends
CPIH 20120314113934 SNIFFER : *** End ***
```

Summary



- To create a service provider or requester in CICS:
 - Create a PIPELINE resource and pipeline configuration file.
 - *Provider only*: create a TCPIP SERVICE resource.
 - Use CICS web service assistant or RDz to create wsbind (and WSDL) files. You will need a COBOL copybook (or other language structure) or a WSDL file.
 - Install the PIPELINE (or issue a PIPELINE SCAN command if already installed).
- Consider Service Flow Modeler for applications that do not have separate presentation and business logic structures.
- Add a sniffer program to the pipeline to diagnose problems.