IBM – CICS Integration Workshop

L20 - CICS Web Services using the CICS Web Services Assistant

Lab Version V61.03.zVA

June 28, 2024

Please send any comments on this lab exercise to:
Steve Fowlkes, Eric Higgins, or Leigh Compton
fowlkes@us.ibm.com
erichiggins@us.ibm.com
lcompton@us.ibm.com

Overview

CICS provides a facility called the Web Services Assistant to generate Web Services definitions (WSDL) from supplied program language structures, and to generate language structures from supplied WSDL documents. The Web Services Assistant is two utility programs that run in z/OS batch. Both programs use the IBM Java SDK.

The purpose of this exercise is to illustrate the use of the CICS Web Services Assistant.

Each section of this lab exercise goes into detail on the various functions of the CICS Web Services Assistant.

Scenario

In our scenario, you are a programmer at a mutual fund company. You have an existing COBOL program called FUNDPROG that can Create, Read, Update, and Delete funds. You want to expose this program as a web service. To quickly expose the program as a web service, you will run the DFHLS2WS program (part of the CICS Web Services Assistant) and generate a WSDL file and a WSBind file.

After installing the new CICS resources, you will test your new Web service.

Lab Requirements

Please note that there are often several ways to perform functions in and for CICS. This lab exercise will present one of the ways. If you are familiar with CICS, you will notice that some of the statements are general, and not necessarily true for every situation.

This lab uses the CICS Explorer, ISPF under TSO, and CICS. If you are not familiar with these, please contact one of the lab instructors for assistance.

The following are other assumptions made in this lab exercise.

- **Any supported version of CICS TS:** This lab exercise uses facilities and naming conventions that were introduced in earlier releases of CICS. Each team performing the lab exercise has their own CICS region, to include their own CSD and other supporting CICS files.
- **Login:** TSO userids are available with appropriate passwords.
- **CICS Explorer:** We will use the latest CICS Explorer, but it is not required for using the CICS Web Services Assistant.

Lab Step Overview

Part 1: Configuring the CICS Explorer connection to CICS

Although you can finish the lab without using the CICS explorer, in this part of the lab, you will configure the CICS Explorer.

Part 2: Understand the Existing Program Interface

In this part of the lab, you will explore the interface of the CICS COBOL program we are working with.

Part 3: Run the DFHLS2WS Program

This part of the lab exercise has you run the DFHLS2WS program. The inputs to the DFHLS2WS program are a Language Structure data definition and input parameters. The outputs are WSDL and a WSBind file.

Part 4: Define a VSAM file and prepare CICS resource definitions

In this step you will define a VSAM file, then define and install the program and file resources in CICS.

Part 5: Install the CICS Web Services Resources

In a previous lab exercise, you may have defined a PIPELINE definition. In this lab exercise we will place a generated WSBind file in the pickup directory of your PIPLINE definition and tell CICS to SCAN the pipeline.

Part 6: Test your new Web Service

We will use the Web Services explorer to test your new Web service.

Part 7: Summary

This is a recap of the steps performed in this lab exercise.

Part 1: Configure the CICS Explorer Connection to CICS

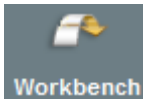
In this part of the lab exercise you will configure the connection between the CICS Explorer running on your workstation to CICS running on z/OS.

Start the CICS Explorer

___1. From the **desktop**, **double-click** the **IBM Explorer** icon to start the CICS Explorer if it is not already running.

___2. **If** prompted for a workspace, from the **Select a workspace** dialog, click the **OK** button to select the default.

___3. If the CICS Explorer shows you a **Welcome page** click the **Workbench icon** in the upper-right corner to go to the workbench. Then **maximize** the window.



Verify that you have connections to z/OS in your CICS Explorer

___4. **This step has been done… Skip**. If you have not already created connections to the z/OS host system, follow the instructions in the Connection Document and then return here. Both the **Remote System Explorer** and **CMCI** connections should be started and active.

Part 2: Understand Existing Program Interface

CICS TS provides the Web Services Assistant to generate Web Services definitions (WSDL) from supplied program language structures, and for generating language structures from supplied WSDL documents. The CICS Web Services assistant is actually two utility programs that run in z/OS batch. Both programs use the IBM Java SDK.

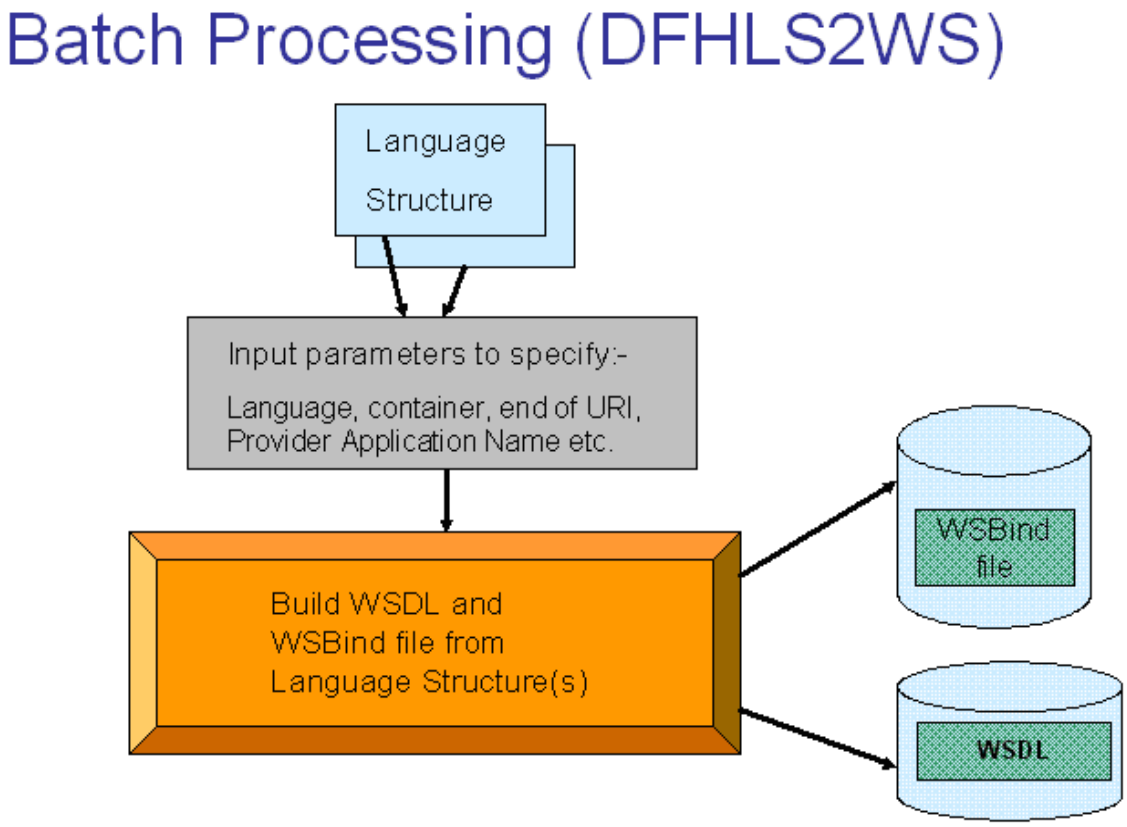
In this part of the lab exercise we will use the DFHLS2WS utility (Language Structure to Web Service) to generate a new WSDL file from an existing data definition. We will be using a COBOL program, but the CICS Web Services Assistant supports COBOL, PL/I, C, and C++. As input, the DFHLS2WS utility needs to import the language copybooks that matches the program's COMMAREA for the request and response. The DFHLSWS utility can invoke a CICS program using the channel interface, to support greater than 32k messages, but for this lab exercise, we will use a COMMAREA interface.

Below is the COBOL definition for the COMMAREA in our FUNDPROG program. This is the layout of the input and output messages for our new Web service.

```
*     FUNDPROG COMMAREA LAYOUT
      03  REQUEST-TYPE              PIC X.
      03  RET-CODE                  PIC XX.
      03  FUND-ID                   PIC X(8).
      03  FUND-NAME                 PIC X(50).
      03  FUND-RATING               PIC X.
      03  FUND-PRICE                PIC X(15).
      03  RETURN-COMMENT            PIC X(50).
```

Note that the DFHLS2WS utility does not fully support the REDEFINES statement since WSDL does not support this concept. This means that if **your** COMMAREA contains a REDEFINES, you will have to specify one of the definitions (i.e. the original or redefined area) as input to DFHLS2WS. **Note**: REDEFINES are supported if it is the last definition in the structure.

Part 3: Run the program DFHLS2WS



For an existing CICS application, the COMMAREA will already be mapped by a language structure. The language structure and a set of input parameters are used as input to a CICS supplied utility which runs as a batch procedure. The procedure is called DFHLS2WS. This converts the supplied language structure into a WSDL document and also generates a WSBind file and a log file.

The **DFHLS2WS** input parameters are:

REQMEM (the definition of the request),
RESPMEM (the definition of the response),
PDSLIB (PDS containing REQMEM and RESPMEM),
LANG (language),
PGMNAME (name of the target program),
PGMINT (program interface (COMMAREA or CHANNEL)),
URI – the name of the web service end-point that will be placed in the WSDL and also used by CICS to dynamically create a URIMAP.
WSBind, **WSDL**, **LOGFILE** – to specify the files to generate for the WSBind file, the WSDL file, and the log file.

The CICS supplied sample JCL to run DFHLS2WS is available on the HFS in the file <CICS_home>/samples/webservices/JCL/LS2WS, which is shown below:

```
//LS2WS1   JOB (MYSYS,AUSER),MSGCLASS=H,
//             CLASS=A,NOTIFY=&SYSUID,REGION=0M
//*
//LS2WS     EXEC DFHLS2WS,
//INPUT.SYSUT1 DD *
LOGFILE=/u/webservices/wsbind/myservice1.log
PDSLIB=//MYHLQ.COPYLIB
REQMEM=MYCOPYBK
RESPMEM=MYCPYBK2
LANG=COBOL
PGMNAME=MYPROG
URI=the/url/to/access/the/service
PGMINT=COMMAREA
WSBIND=/u/webservices/wsbind/myservice1.wsbind
WSDL=/u/webservices/wsdl/myservice1.wsdl
*/
```

For a full description of the input parameters, search for the topic “DFHLS2WS: high level language to WSDL conversion” in the CICS Documentation.

This JCL has been customized for the workshop environment as follows (note that in addition to names and locations for our environment we specified mapping level 4.0 (it is recommended to use the most current mapping level):

```
->JOB Cards here<-
//*
//  SET QT=''''
//*
//LS2WS    EXEC DFHLS2WS,
// TMPFILE=&QT.&SYSUID.&QT.
//INPUT.SYSUT1 DD *
LOGFILE=/u/user1/logs/fundProg.log
PDSLIB=//USER1.CICSLAB.UTIL
REQMEM=FUNDCOMM
RESPMEM=FUNDCOMM
LANG=COBOL
MAPPING-LEVEL=4.0
PGMNAME=FUNDPROG
URI=cicslab/fundProg
PGMINT=COMMAREA
WSBIND=/u/user1/cicslab/wsbind/fundProg.wsbind
WSDL=/u/user1/cicslab/wsdl/fundProg.wsdl
*/
```

Log generated file

The log file contains detailed information on the steps the job has taken to create the output files. The contents of the log file are not normally required by end users. However, it does contain a copy of the generated WSDL which can be used if the generated WSDL file is misplaced. The log file could also be used in problem determination.

WSDL generated file
The WSDL file defines all the information required for a client to access this Web service. It contains an XML schema representation of the request and response and location information for the service. As generated using the statements above, the WSDL file needs to be customized with the correct TCP/IP address and port before it can be published to clients. Using the more recent mapping levels, the full scheme, host, port, and path can be specified in the input parameters so the resulting WSDL does not need to be modified.

WSBind generated file
The WSBind file contains the meta-data that CICS uses at runtime to marshal and de-marshal the request and response messages in XML into the format expected by the application program. The WSBind file is read by CICS when a WEBSERVICE resource is installed onto a PIPELINE (it is not read on every request).

It is important to ensure that the version of the WSBind file matches the WSDL file used by clients. If changes are made to the language structure, the WSDL and WSBind file need to be re-generated, and the new WSDL sent to the client.

If the PIPELINE scanning mechanism is used to install a WEBSERVICE, the resource name will be based on the name of the WSBind file. It is suggested to keep the names of the WSBind file and the WSDL file the same to ensure installed Web services and clients match.

Create a copybook for the Commarea

CICS Web Services Assistant requires that the COBOL structures defining the input and output for the program be in a file (or files) separate from the program itself. The target program we will be using simply defines the COMMAREA layout in line rather than including it with a COPY statement. If your program is similarly coded, you would need to create a separate COBOL source file (i.e. a copybook) with just the structure of the Commarea.

For this lab, your instructors have created member FUNDCOMM for you in **USER1.CICSLAB.UTIL**. The member looks like this, with the 03 beginning in area B (column 12).

```
03  REQUEST-TYPE            PIC X.
03  RET-CODE                PIC XX.
03  FUND-ID                 PIC X(8).
03  FUND-NAME               PIC X(50).
03  FUND-RATING             PIC X.
03  FUND-PRICE              PIC X(15).
03  RETURN-COMMENT          PIC X(50).
```

Compile FUNDPROG

___1. From the **Remote System Explorer** perspective, the **Remote Systems** view, expand the **USER1.CICSLAB.JCL** file, **right-click** the **FUNDPROG** member, and from the context menu, select **Submit.**

___2. From the **Job Submission Confirmation** dialog, click the **OK** (or **Locate Job**) button.

___3. From the **Remote System Explorer** perspective, the **JES** node, verify that your job ran **successfully**. **Note** that you may have to refresh the view. **Note** that the return codes are towards the top of the JCL listing.

Submit the LS2WSJCL

___4. From the **Remote System Explorer** perspective, the **Remote Systems** view, expand the **USER1.CICSLAB.JCL** file, **right-click** the **LS2WSJCL** member, and from the context menu, select **Submit**. Then from the **Job Submission Confirmation** dialog, click the **OK** button (you may click the **Locate Job** button).

___5. From the **Remote System Explorer** perspective, the **JES** node, **My Jobs**, **double-click** your Job (you may need to refresh the job listings).

___6. **Ensure** that your LS2WS **Job** ran **successful**ly. **Note** step return codes are towards the top of the listing. They should all be 0. Close the Job Listing when you are done looking at it.

Look at the LS2WSJCL file

___7. From the CICS Explorer, the **Remote Systems** view, double-click **USER1.CICSLAB.JCL(LS2WSJCL)** to open it in an editor. (this is the JCL just used to generate the WSDL and a WSBind file)

The input statements indicate that:
The PDS containing the members describing the input and output messages are in USER1.CICSLAB.UTIL.

The PDS member describing the request (REQMEM) is named FUNDCOMM.

The PDS member describing the response (RESPMEM) is name FUNDCOMM.

The program interface (PGMINT) is a COMMAREA.

The application program (PGMNAME) that will be invoked after the COMMAREA is prepared (conversion from XML to COMMAREA is complete) is named FUNDPROG.

___8. **Close** the file.

___9. From the **Remote System Explorer** perspective, the **z/OS UNIX Files** node, navigate to the **MyzOS > z/OS UNIX Files > My Home > cicslab > wspickup > provider** directory.

The LS2WSJCL job you just ran created the WSBind file and WSDL file for your Web service. Based on the input parameters, the WSBind file was placed in

/u/user1/cicslab/wspickup/provider and is called fundProg.wsbind.

The WSDL file was also placed in

/u/user1/cicslab/wspickup/provider and is called fundProg.wsdl.

Note: You may need to click the refresh button on the z/OS UNIX Files view to see your new WSBind and WSDL files. If you can not see the files contact a lab instructor.

Part 4: Define a VSAM file and add CICS resource definitions

The FUNDPROG program needs a VSAM file named USER1.CICSLAB.FUNDFILE. In this step we will define the VSAM file that the application program needs and add resource definitions in CICS for the program and file.

Submit the FUNDVSAM job to create the VSAM file

___1. From the **Remote System Explorer** perspective, the **Remote Systems** view, expand the **USER1.CICSLAB.JCL** file, **right-click** the **FUNDVSAM** member, and from the context menu, select **Submit**. Then from the **Job Submission Confirmation** dialog, click the **OK** button.

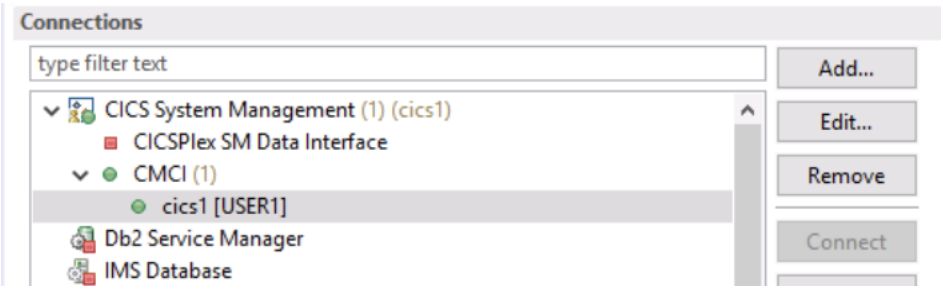
___2. From the **Remote System Explorer** perspective, the **JES** node, click the **refresh** button, then **double-click** your **Job**.

___3. **Ensure** that your FUNDVSAM **Job** ran **successful**ly. **Note** step return codes are towards the top of the listing. They should all be 0. **Close** the Job Listing when you are done looking at it.

Change to the CICS SM perspective

___4. For the next few steps, you will need to be in the **CICS SM** perspective. If not already open, select this perspective by **selecti**ng **Window > Perspective > Open Perspective > Other > CICS SM**.

___5. Verify that the CICS Explorer is connected to your CICS region. In the Host Connections view, **expand CICS System Management > CMCI**. If the CICS Explorer is connected to CICS, the CICSZ entry will have a green dot. If you see a red dot next to CICSZ, select it with the cursor and **click** the **Connect** button.



Verify the TCPIPService definition

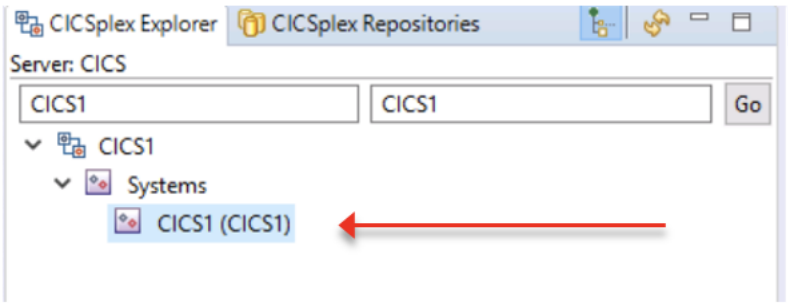
___6. A **TCPIPService** resource definition is needed to tell CICS to listen for incoming HTTP requests. **We have added a TCPIPService definition for you** name DFH$WUTC, but **if** one did not exist, you would need to create one with the characteristics similar to those in the following table. Note that the TCPIPService definition we defined for you does not require security over the connection (SSL or HTTP Basic Authentication).

Field	Value
TCPIPService	HTTPPORT (some name)
Group	WORKSHOP
POrtnumber	1423 (or the port you have been assigned>
PROtocol	Http
Transaction	CWXN

___7. Note that the TCPIPService definition we added causes the CICS region to listen on port 1423. For this series of lab exercises, we are using the same port for all HTTP-based requests. In your environment you may be using different ports for Web services and browser requests depending on your application design, firewall, network configuration, etc.

Define and install a PIPELINE definition

___8. From the CICS Explorer, the **CICS SM** perspective, from the **CICSplex Explorer** view, **select your CICS region**.



___9. From the **CICS SM** perspective, from the **menu bar**, select **Definitions > Pipeline Definitions**. From the **Pipeline Definitions** view, **right-click** in an **open area** and from the context menu select **New**.

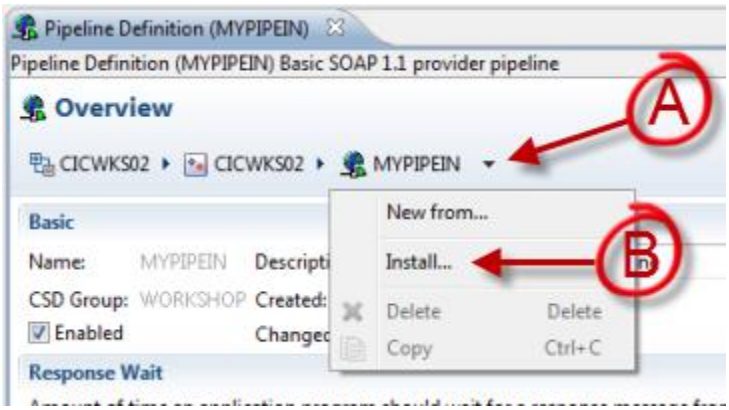
___10. From the **Create Pipeline Definition** pop-up dialog, **define** a new **PIPELINE** definition with the following attributes. **After** you fill in the details from the first table, click the **Finish** button to open the PIPELINE resource in an editor. **Modify** the resource definition as indicated in the second table. Note: this definition is case sensitive.

Attribute	Value
Group	WORKSHOP
Name	MYPIPEIN
Description	Basic SOAP 1.1 provider pipeline
Configfile	/u/user1/cicsProvider.xml
Open editor	Checked

In the **Pipeline Definition (MYPIPEIN)** editor, from the **Overview tab** (across the bottom), **enter the following**. Be sure that your changes took effect, and **save** MYPIPEIN.

Attribute	Value
Shelf	/u/user1/cicslab/shelf/
WSBind Pickup Directory	/u/user1/cicslab/wspickup/provider/

___11. To **Install** the PIPELINE resource, first **save** the resource (press Ctrl-S) and then click MYPIPEIN in the A section (see diagram below), then click **Install**. From the **Perform INSTALL Operation** dialog, **highlight** your **CICS region**, then click **OK**.



Close the **Pipeline Definition (MYPIPEIN)** editor.

Define and install a PROGRAM definition

___12. From the CICS Explorer, the **CICS SM** view, the **menu bar**, select **Definitions > Program Definitions**. From the **Program Definitions** view, **right-click** in an open area and select **New**. Then **define** and **install** the following program definition. (This may already be created if you have already performed other the exercises using the FUNDPROG program.)

Attribute	Value
Group	WORKSHOP
Program	FUNDPROG
Language	Assembler, C/C++, COBOL, or PL/1

Define and install a FILE definition

___13. From the CICS Explorer, the **CICS SM** view, the **menu bar**, select **Definitions > File Definitions**. From the **File Definitions** view, **right-click** in an open area and select **New**. Then **define, save,** and **install** a file definition for the **VSAM file** named **FUND**.

Attribute	Value
Group	WORKSHOP
Name	FUND
Description	The FUNDFILE file
DSNAme	USER1.CICSLAB.FUNDFILE
Open editor	Checked

From the **File Definition (FUND)** editor, **click** the **Attributes** tab (at the bottom).

Attribute	Value	
Record Format	Fixed	(use the pull-down menu to select “Fixed”)
Add	Yes	(use the pull-down menu to select “YES”)
Browse	Yes	(use the pull-down menu to select “YES”)
Delete	Yes	(use the pull-down menu to select “YES”)
Read	Yes	
Update	Yes	(use the pull-down menu to select “YES”)

Check for the message at the bottom of the Explorer; CNX0609I: Install of File Definition "FUND" into CICS1 succeeded.

You have just created a VSAM file, and defined the pipeline, program, and file to CICS and installed the definitions.

Part 5: Install the CICS Web Services Resources

When you ran the DFHLS2WS utility, you asked for the fundProg.wsbind file to be placed in /u/user1/cicslab/wspickup/provider. This directory was specified as the WSDir attribute of the PIPELINE definition you just created. WSDir is often referred to as the "pickup" directory since the Pipeline process will pick up any WSBind files which have been placed in it. This pickup happens automatically when the PIPELINE resource is installed. You can also tell CICS to SCAN the directory for any new and changed WSBind files. Any new or changed files will be used to create WEBSERVICE and URIMAP resources.

Note that if the Web service's WSDL is in the pickup directory (with the same name as the WSBind file), CICS will create a URIMAP so that you can access the WSDL using the common convention of indicating the Web service endpoint (path) with "?wsdl" as a query string on the end of the URL. When you created the WSDL, you told the DFHLS2WS utility to put the WSDL into the /u/user1/cicslab/wspickup/provider directory.

___1. If you installed the PIPELINE resource in **Step 9** above, skip to **Step 3**. The install of the PIPELINE resource caused an implicit SCAN of the Pipeline's pickup directory.

___2. From the CICS Explorer, the **CICS SM** perspective, from the **menu bar** select **Operations > Pipelines**, then **right-click** on the **MYPIPEIN** pipeline definition and from the context menu, select **Scan**. From the **Perform SCAN Operation** dialog, click **OK**.

Note that the SCAN just asked CICS to start a scan. The response you saw when the window went away was successful. A successful response just indicated that the start of the scan was normal. It didn't indicate that all the work you wanted the scan to complete was successful. This means that although you received a successful response, your fundProg Web service may not be usable.

___3. From the CICS Explorer, the **CICS SM** perspective, from the **menu bar**, select **Operations > URI Maps**. From the **URI Maps view**, verify that your new Web service has been installed properly. Since the entries were dynamically generated, the Name will start with a '$'.

NOTE that because you put the WSDL for your Web service in your PIPELINE's pickup directory, and because the WSDL has the same name as the WSBind file, CICS has added a URIMAP with the same path as your Web service with "?wsdl" at the end of the URL. If this path is requested, CICS will provide the WSDL for the Web service. This is a common convention.

___4. You can use the **CICS SM** perspective in CICS Explorer to view your CICS **Web Service** to verify your web service was properly installed (Operations > Web Services).

You have just exposed your CICS COBOL Program as a Web service.

Part 6: Test your Work

We will be testing with the **Web Services Explorer** in the CICS Explorer using the WSDL you generated from the CICS Web Services Assistant. In this part of the lab exercise you will verify that CICS has made the WSDL accessible via an HTTP URL. You will also copy your WSDL from z/OS (where you generated it using the CICS Web Services Assistant) via CICS, to your workstation (where you will use it with the Web Services Explorer). You will then use the Web Services Explorer to invoke your CICS-based Web service.

___1. Open a web browser. On the address line type **http://wg31:1423/cicslab/fundProg?wsdl** and press the **enter** key. You should see your web service's WSDL.

The web browser isn't showing you an exact representation of the WSDL but is interpreting the XML. Since we need a copy of the WSDL file, using a right-click in the middle of the web page and selecting 'view page source; from the context will bring up the actual WSDL file. Note that the wording on the context menu varies slightly between Web browsers; in Internet Explorer it is 'view source'. You could then copy and paste the WSDL into a file.

Note that CICS TS has dynamically added a URIMAP resource to make this possible. You could manually create and install a URIMAP definition to achieve the same. If you manually add the URIMAP, be sure to indicate that CICS should convert the file to ISO-8859-1 (the defacto standard code page for web browsers) or UTF-8 depending on your requirements.

___2. From the **browser window**, **right-click** in the middle of the window, and from the **context menu**, select '**View Page Source**'. We will be copying the WSDL generated by the CICS Web Services Assistant from z/OS to your workstation.

___3. On the window that just opened, press **Ctrl-A** to mark all of the WSDL source, and **Ctrl-C** to copy it to a clipboard.

___4. In the CICS Explorer, open (or switch to) a **Resource** perspective.

___5. From the **Project Explorer** view, right-click in an open area and from the context menu select **New > Project.**

___6. From the **New Project** dialog, **expand General** and **select Project**. Click the **Next** button.

___7. From the **Project** dialog, specify a **Project name** of **LS2WSLab**, and click the **Finish** button.

___8. From the **Project Explorer** view, **right-click** on your **LS2WSLab** project, and from the context menu, select **New > File**.

___9. From the **New File** dialog, specify **fundProg.wsdl**, and click the **Finish** button.

___**10.** The edit view should open; if not **Double-Click** your **fundProg.wsdl** file to open it in the **editor**.

___**11.** From the **fundProg.wsdl** editor, **click** the **Source tab** (along the bottom), click in the fundProg.wsdl editor, then press **Ctrl-V** to paste the WSDL into the editor.

___**12.** **Save** and **Close** the fundProg.wsdl editor.

___**13.** From the **menu bar**, select **Window > Preferences**, on the **left** expand **General** and click **Web Browser**. On the **right** click the **radio button** next to **Use external web browser**, and **click** the **box** next to **Firefox**. Then click the **Apply and Close** button at the bottom right **to dismiss** the Preferences **editor**.

___**14.** From the **Project Explorer** view, **expand** your **LS2WSLab** project, right-click on **LS2WSLab > fundProg.wsdl**, and from the context menu, select **Web Services > Test with Web Services Explorer**. You will notice that a Web Services Explorer view opens.

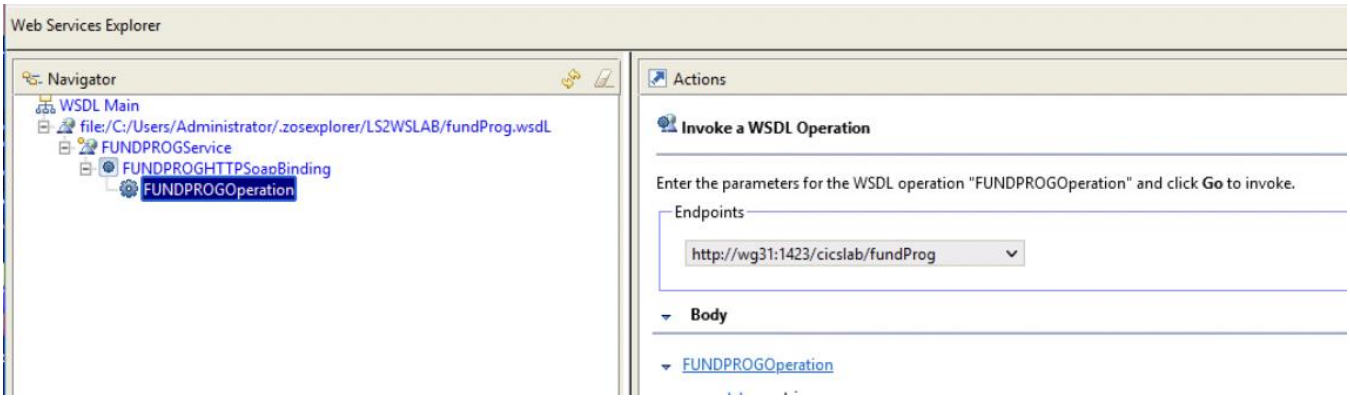
___**15.** In the **Navigator** pane, click on **FUNDPROGHTTPSoapBinding**.

___**16.** In the **Actions** pane, in the **Endpoints** section, click the **Add** hyperlink.

___**17.** **Modify** the added line to specify the host and port for your CICS region. In this lab, the new endpoint would be **http://wg31:1423/cicslab/fundProg**

___**18.** In the **Actions** pane, click the **Go** button.

___**19.** In the **Navigator** pane expand **FUNDPROGHTTPSoapBinding**, then click **FUNDPROGOperation** (the operation name). You should see a window similar to the following.

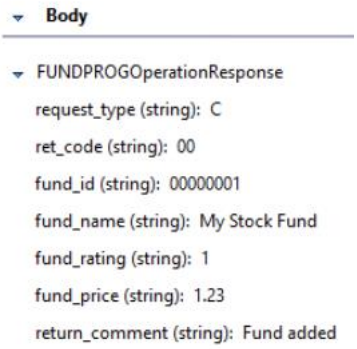


___**20.** Note in the **Actions** pane, the Endpoints field should show the new endpoint.

___21. Type in the following values and click the **Go** button of the Actions pane.

Field	Value
request_type	C (for Create)
ret_code	00
Fund_id	00000001 (8 characters)
Fund_name	My Stock Fund
Fund_rating	1
Fund_price	1.23
return_comment	(blank)

___22. This will cause the FUNDPROG program to add a record in the FUNDFILE VSAM file. In the **Status pane**, you should see something similar to the following. **Click** the **Source** link in the upper right corner of the Status window to change formatting.



___23. Hmm – what if your test wasn't successful?

Your approach to debugging a problem when using CICS-based Web service will be similar to the debugging techniques you already use for other CICS programs. Some of the areas you might check, depending on where you think the problem is, may be:

- Did the request get to CICS? If you have no network connectivity, or CICS isn't listening, you may get a connection refused.
- Is the Web service defined to CICS? If you didn't specify the right path (cicslab/fundProg in this case), CICS won't know what to do with the request and will return an HTTP 500 response code. Likewise, if the cicslab/fundProg Web service hasn't been properly defined to CICS (like maybe your SCAN didn't do what it was supposed to, or maybe your PIPELINE definition didn't get properly installed), then again, CICS won't know what to do with the request and will return an HTTP 500 response code.
- Was your target business logic program invoked? You could check the program count on your program to check if it was invoked. If your program is being invoked, then it is business as usual and you can use your favorite debugger to step through your program.
- You can also use CEDX to go through your program. By default the transaction we are running our Web service under is CPIH, so you can 'CEDX CPIH' to walk your program through EDF.

- If you received a ‘return_comment: Problem adding fund, see CICS message log”, you may have defined the file incorrectly. It should be Fixed length, readable, addable, updateable, and browseable.

Contact a lab instructor if you have any questions.

Part 7: Summary

Congratulations, you have run the DFHLS2WS utility and exposed a CICS COBOL program as a Web service.

In this lab you:

- Ran the DFHLS2WS utility using the COBOL program's COMMAREA as input
- Compiled the CICS COBOL program, created a VSAM file.
- Defined and installed the program and file resource definitions in CICS
- Defined and installed a pipeline resource definition in CICS
- Told CICS to pick up your new Web service from the WSBind file placed in the pickup directory
- Tested your new Web service

If you have any questions about anything you did in this lab exercise, please contact one of the lab instructors.