

IBM TechU

2021 Edition



IBM

Lab Exercise: **Using IBM zUnit to Unit Test a COBOL/CICS/DB2 program using Local assets.**

Session ID: [z204709](#)

Speaker Name: [Wilbert Kho](#)

Room: [?????...](#)

FOR LAB ROOM ONLY -- DO NOT REMOVE, DEFACE or WRITE ON THIS LAB BOOK

Introduction

This lab will take you through the steps of using the automated unit testing (**zUnit**) capabilities of IBM Developer for z (**IDz**) to create a unit test case for a **COBOL CICS/DB2 program**. This enables a developer to test early without impacting other developers that share the same CICS environment and using batch jobs. In this lab you will record interaction with a COBOL CICS/DB2 program and run a test case where we add a bug in the program to be tested.

Instructions to use the Windows + Linux + Z/OS on Cloud

There are 2 ways to access the Windows + z/OS + Linux cloud instances..

1. Via **Windows Remote Desktop** (Better performance, but firewalls might block)
2. Via **Web Browser** (suggest Firefox or Chrome)

In any case be sure that you have the **provided link to access the cloud instance**, the userid and **password** to access the Windows client remotely.

Please always **Use COPY/PASTE for password**.

Some letters may be identical, example: **I** (uppercase i) and **l** (lowercase L).

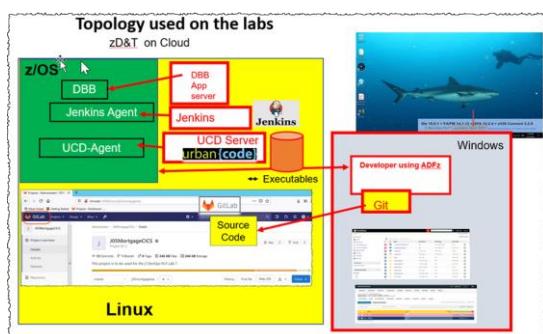
Example of provided links

| Serial No | URL | User ID for Web browser | IP Address for Remote Password | User ID for Remote Desktop | Domain | machine # |
|-----------|---|-------------------------|--------------------------------|----------------------------|-------------------------|-----------|
| 1 | https://169-63-141-246-T-5282.ibmtrialmachines.com/ | Administrator | 169.63.141.246 | K58E6Pep | Wwin-5282\Administrator | 5282 |
| 2 | https://169-60-90-69-T-5283.ibmtrialmachines.com/ | Administrator | 169.60.90.69 | EP4tmGLS | Wwin-5283\Administrator | 5283 |

Below is what you will access using the cloud environment.

Be sure that you have an IP address, the userid (**Userid will be different if using Remote**

Desktop or Browser) and password to access the Windows client remotely.



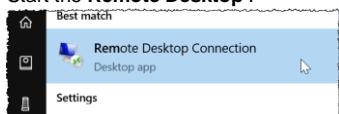
Using Windows Remote Desktop (preferable way)

This is the preferable way to run the labs, but in some customer location this capability is blocked via Firewalls. If this is your case, use the Web Browser. Instructions are listed here as well.

If you are a Mac user may consider downloading the Remote desktop from
<https://itunes.apple.com/us/app/microsoft-remote-desktop-10/id1295203466?mt=12>

1. Here one example using Windows 10.

Start the **Remote Desktop**.

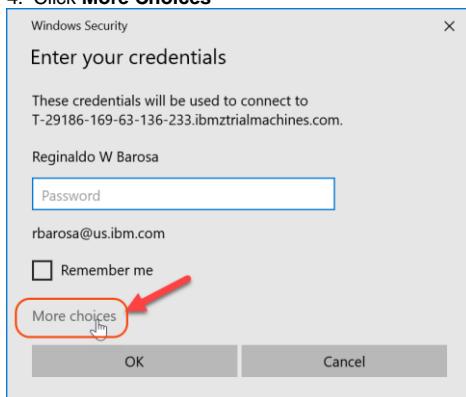


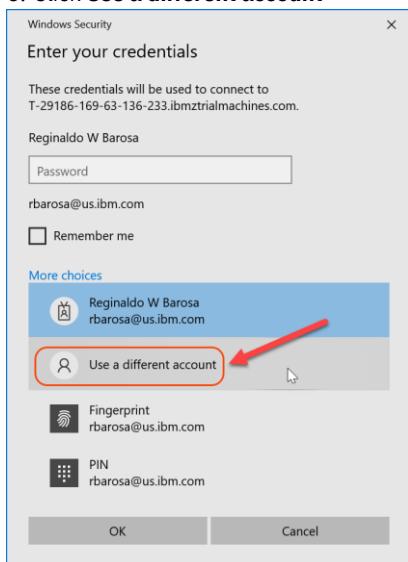
2. Click **Connect**



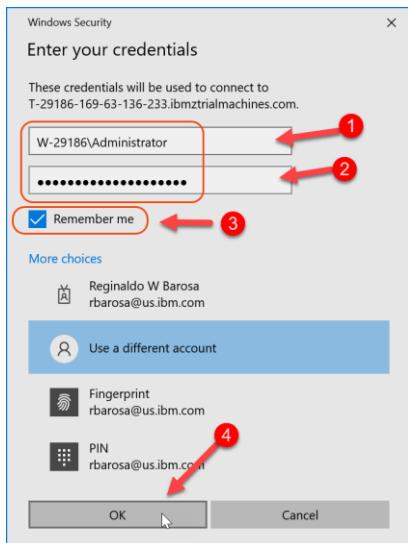
3. Click **Connect** if a dialog as to *trust the remote connection*

4. Click **More Choices**



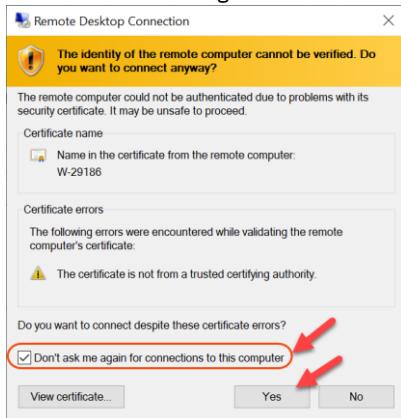
5. Click Use a different account

6. Copy and paste the provided “**User ID for Remote Desktop**” and **Password**, select **Remember me** and click **OK**.

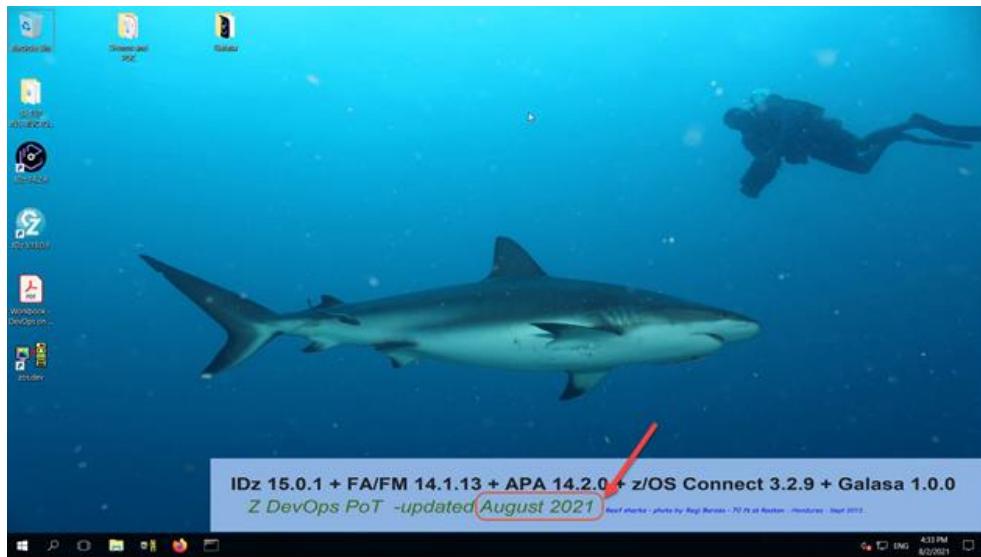


7. You should get a dialog as below.

Select Don't ask me again for connections to this computer and click Yes



8. You should get a screen with a shark. That indicate that you have access to the Windows client on cloud.



On the windows desktop there is a PDF icon with the Labs workbook.. you will find this lab also there.
It may help for copy/paste, etc...

But to follow the labs I suggest printing or use another display or iPhone/iPad to better follow the instructions or use another Monitor to follow the lab instructions.

Using Web Browser (if using VPN or Firewalls that block Remote Desktop)

Commented [RWB1]:

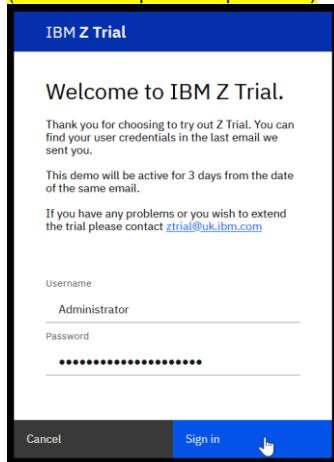
Be sure that you have the **provided link to access the cloud instance**, the userid (**Administrator**) and **password** to access the Windows client remotely. Always use **COPY/PASTE** for password.

Some letters may be identical, example: **I** (uppercase i) and **l** (lowercase L).

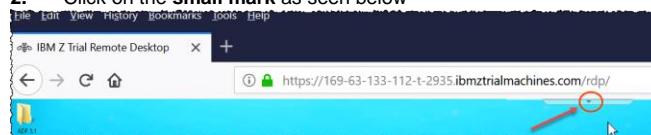
Example

| jwIBM Z System DevOps Workshop | | User ID for Web browser | IP Address for Remote Password | User ID for Remote Desktop | Domain | machine # |
|--------------------------------|---|-------------------------|--------------------------------|----------------------------|-------------------------|-----------|
| Serial No | URL | Administrator | 169.63.141.246 | K58E6Pep | Wwin-5282\Administrator | Wwin-528 |
| | 1 https://169-63-141-246-t-5282.ibmtrialmachines.com/ | Administrator | 169.60.90.69 | FRBtmG15 | Wwin-5283\Administrator | Wwin-5283 |
| | 2 https://169-60-90-69-t-5283.ibmtrialmachines.com/ | | | | | |

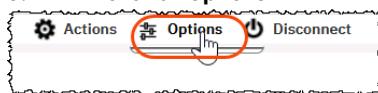
1. Use the link provided by the instructor and type **Administrator/password** provided and click **Sign in** (Remember to paste the password)



2. Click on the **small mark** as seen below



3. Click on **Options**

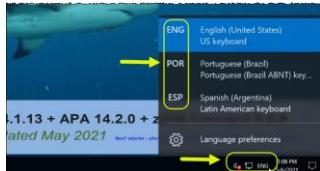


And select **Full Screen**

Adjusting the keyboard for other languages than English

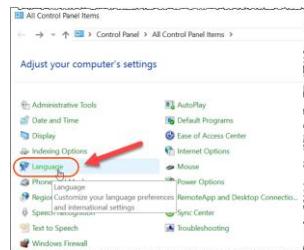
In some countries the keyboard must be mapped due language differences.

If you are in Brasil or any other Latin America country can select the language as below

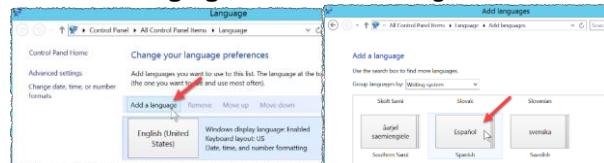


If you use another keyboard other than ENG, POR or ESP, you need to make updates..

1. Go to windows **Control Panel** and select **Language**



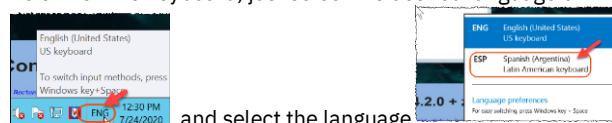
2. Click **Add a Language**, and follow the dialogs. See one example on the screen captures below



Click Add



To switch the keyboard, just select the desired language on the right corner of the screen



and select the language

Exercise instructions

Overview of development tasks

To complete this tutorial, you will perform the following tasks:

1. **Get familiar with the application using the 3270 terminal**
→ You will start a 3270 emulation and execute a transaction named **HCAZ** to become familiar with the Application that you will be recording.
2. **Record interaction with the application.**
→ You will record an interaction with the COBOL CICS/DB2 program.
3. **Generate, build and run the unit test**
→ You will compile and link-edit the generated unit test program, followed by running the unit test.
4. **Introduce a bug in the program and rerun the unit test**
→ You will modify the program under test, rerun the unit test, and observe the failure of the test case.
5. **Run the unit test from a batch JCL .**
→ You will run the unit test from a Batch JCL and observe a similar test case result.

Section 1. Get familiar with the application using the 3270 terminal

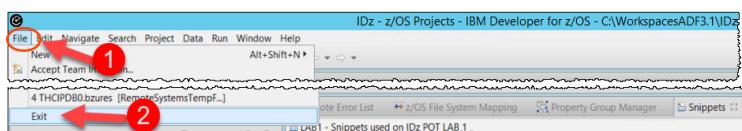
You will start a 3270 emulation and execute a transaction named **HCAZ** to become familiar with the Application that you will be interacting with.

1.1 Connect to z/OS and emulate a CICS 3270 terminal

You will use IDz to emulate a 3270 terminal to run the CICS transaction.

1.1.0 This Lab will use IDz Version 15, and if it is already running jump to step 1.1.2

► If IDz version 14 used on LAB 1 is running, you must close it using **File** and **Exit**.



1.1.1 Start IBM Developer for z Systems version 15

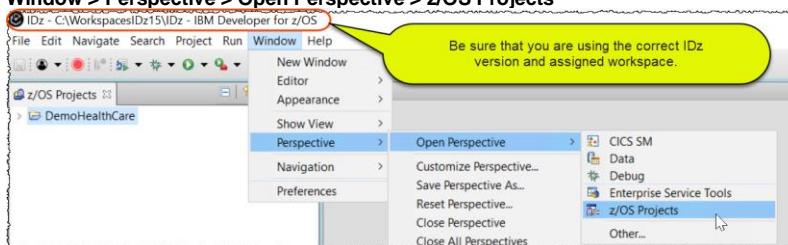
► Using the desktop double click on **IDz V15** icon.

► Verify that the message indicates that it is Version **15.0.1**

**IMPORTANT -> This icon will start an eclipse workspace that has already some definitions required for this lab.
PLEASE DO NOT start IDz using other way than this icon.**



**1.1.2 ► Open the z/OS Projects perspective by selecting
Window > Perspective > Open Perspective > z/OS Projects**

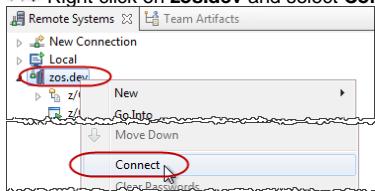


1.1.3 On this lab you will use userid **ibmuser.**

If you are already connected as ibmuser, jump to step 1.1.5 Otherwise disconnect from z/OS and reconnect

► Using **Remote Systems** view, right click on **zos.dev** and select **Disconnect**

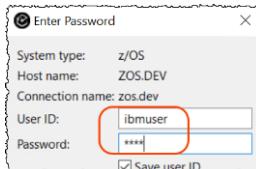
► Right click on **zos.dev** and select **Connect**



1.1.4 ► Type **ibmuser** as userid and **sys1** as password.

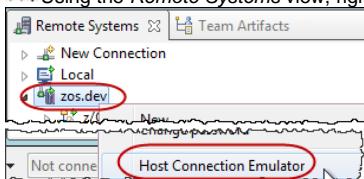
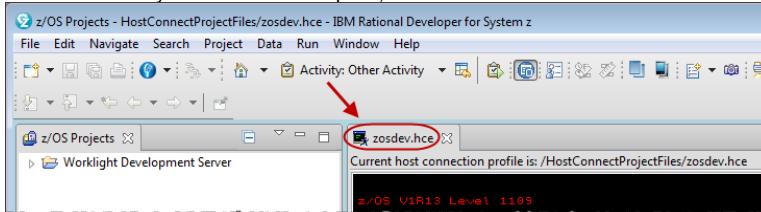
The userid and password can be any case; don't worry about having it in UPPER case.

Click **OK** to connect to z/OS.



1.1.5 Wait until connection is complete (on the bottom and left until the green bar disappears)

► Using the **Remote Systems** view, right click on **zos.dev** and select **Host Connection Emulator**.

1.1.6 ► Since you will need more space, double-click on the **zos.dev.hce** title1.1.7 ► Type **I cicsts54**. (where "I" is the lower case of letter "L") and press **Enter** key.1.1.8 ► Logon using your z/OS user id **ibmuser** and password **sys1** and press **Enter**.

1.1.9 The sign-on message is displayed



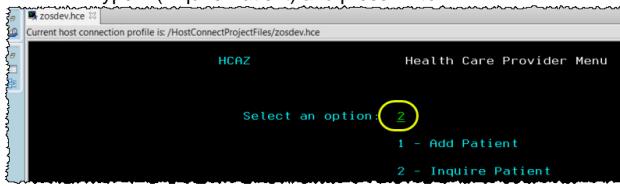
1.2 Run CICS transaction HCAZ

You should now be in the z/OS CICS region named C/CSTS54. This is the CICS instance where you will make the recording of your interaction.

- 1.2.1 ► Type the CICS transaction **hcaz** and press the **Enter** key.



- 1.2.2 ► Type 2 (Inquire Patient) and press Enter .

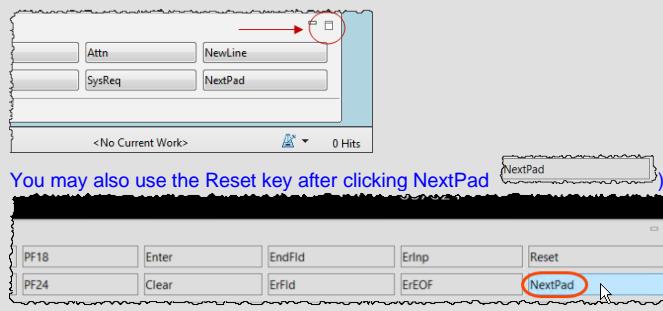


- 1.2.3 ► Type 1 for Patient ID and press Enter. The program will read the patient from a DB2 table and display the customer details. Remember the patient first name: **Ralph**



IF you need to use functions like Clear or Reset

If you need to use the **clear** function use the key **Esc**.
Also you may look in the right lower corner, select this icon □ . This will display possible keys, including the clear button.



1.2.4 ► Press **F3** to end the application.



1.2.5 ► Close the terminal emulation clicking on → . Or pressing **CTRL + Shift + F4**.

1.2.6 ► You may need to restore the z/OS Projects perspective by clicking on the icon on top right:



What have you done so far?

You emulated a 3270 terminal using IBM Developer for System z.
You also executed the CICS transaction **HCAZ** and verified a simple interaction with the Health Care application. The objective here was to show the code that you will update.

Section 2 – Record data interaction using the CICS application.

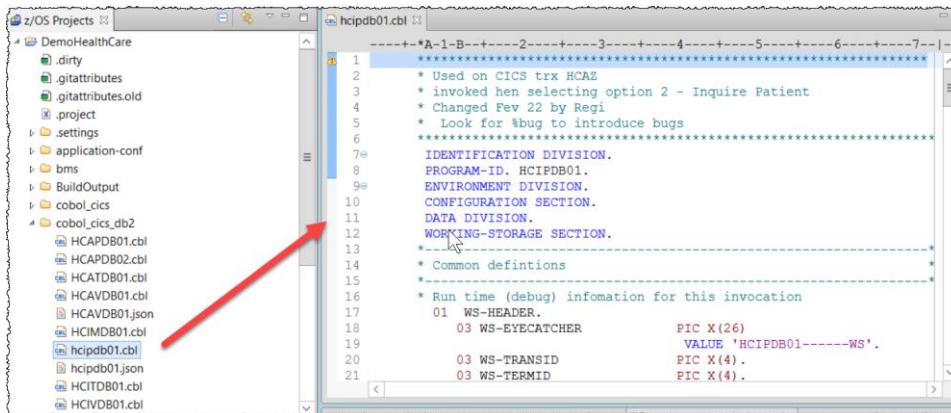
Using IDz you will record the interaction with the COBOL/DB2 program that reads a patient from a DB2 table.

2.1 Understanding the COBOL program that reads from DB2 table

The COBOL code that reads the patient from the DB2 tables is the program **HCIPDB01**

2.1.1 ► Using z/OS Projects view, expand **DemoHealthCare**
double click on **hcipdb01.cbl** under **DemoHealthCare/cobol_cics_db2**.

This is the program that you will update later on. The name is lower case to make it easier to find it 😊 .



2.1.2 Using the Outline view on left expand **PROCEDURE DIVISION** and click on **GET-PATIENT-INFO.**

This is where the patient data is read from the DB2 table.

Later on you will introduce a bug in this program..

The screenshot shows the IBM Rational Application Developer interface. On the left, the 'Properties' view displays the following structure:

- PROGRAM: HCIPDB01
- IDENTIFICATION DIVISION.
- ENVIRONMENT DIVISION.
- DATA DIVISION.
- PROCEDURE DIVISION.
- MAINLINE SECTION.
 - MAINLINE-END.
 - MAINLINE-EXIT.
- GET-PATIENT-INFO.

A red arrow points from the 'GET-PATIENT-INFO.' entry in the properties list to the corresponding code in the main editor window. The main editor window contains the following COBOL code:

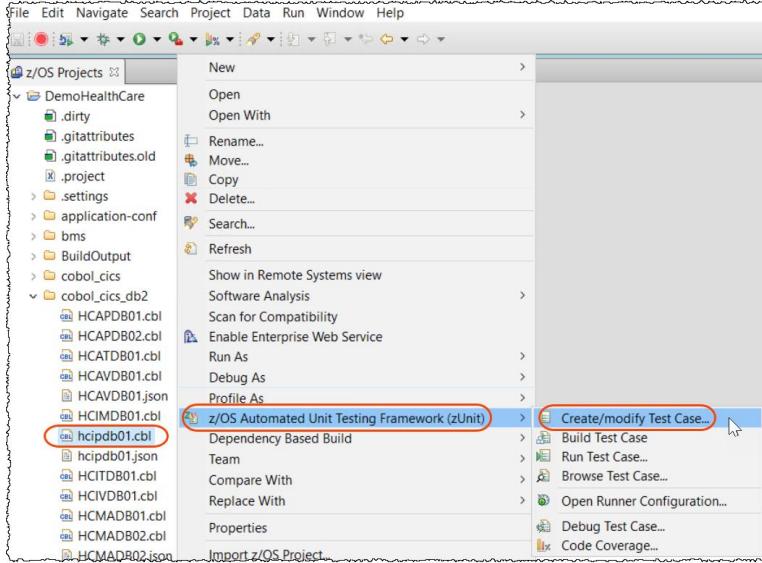
```
MAINLINE-END.  
EXEC CICS RETURN END-EXEC.  
MAINLINE-EXIT.  
EXIT.  
*  
GET-PATIENT-INFO.  
EXEC SQL  
  SELECT FIRSTNAME,  
        LASTNAME,  
        DATEOFBIRTH,  
        insCardNumber,  
        ADDRESS,  
        CITY,  
        POSTCODE,  
        PHONEMOBILE,  
        EMAILADDRESS,  
        USERNAME  
  INTO :CA-FIRST-NAME,  
       :CA-LAST-NAME,  
       :CA-DOB,
```

A red box highlights the 'GET-PATIENT-INFO.' section of the code. The bottom right corner of the red box has a small 'I' icon. The bottom status bar shows tabs for 'Remote Error List', 'z/OS File System Mapping', and 'Property Group Manager'.

2.2 Recording the COBOL program that sends the message

2.2.1 ► To start the recording, right click on **hcipdb01.cbl**

and select z/OS Automated Unit Testing Framework (zUnit) > Create/modify Test Case...

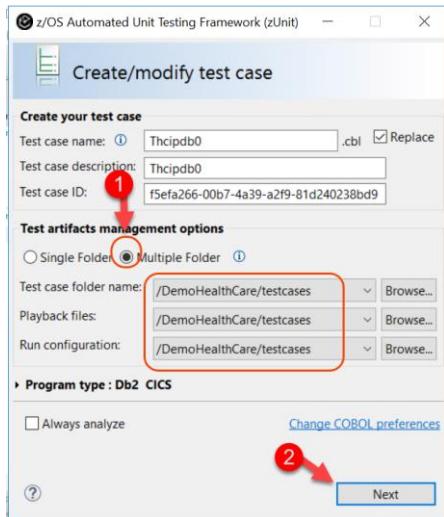


Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

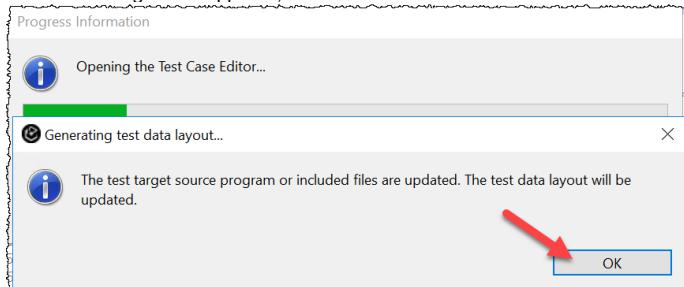
2.2.2 This opens a dialog where you can name your test case.

► Click on **Multiple Folder**, verify that the generated assets will be under **/DemoHealthCare/testcases** and click **Next**.



2.2.3 This operation will generate the test data layout on the local workspace.

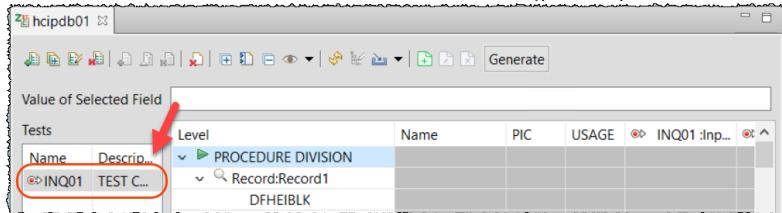
► If the dialog below appears, click **OK**.



2.2.4 This will open the **Test Case Editor**, as shown below. **INQ01** may or may not be on your screen.

If **INQ01** is there you will delete on next step.

The reason that this could be there is because we are reusing a workspace where this was created before.



Understanding the test case editor

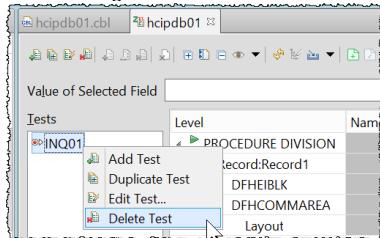


- The bottom left box summarizes all the input output variable structures – In this example, the program has a COMMAREA exchanged via the PROCEDURE DIVISION, five EXEC CICS statements and one EXEC SQL statement.
- The COMMAREA and the CICS statements are also accompanied by the CICS DFHEIBLK variables. To see those values you must select the icon



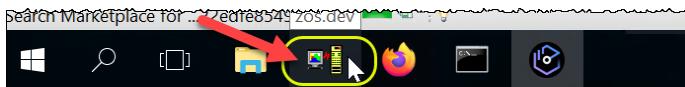
The variables that are returned by the program are the output from the program logic that a developer should be checking

2.2.5 ► Since we will be recording the test data, delete the **INQ01** or any other entry (if it exists) by right clicking and selecting **Delete Test**



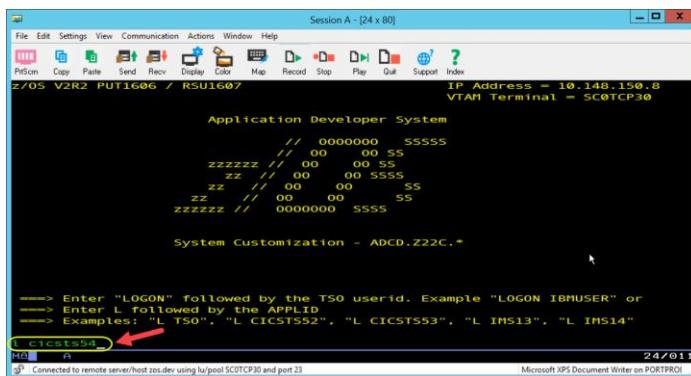
Notice that on section 1 we used the IDz emulator and now you will use the IBM PCOM emulator. Any 3270 emulator can be used.. We just show here a different way to emulate a 3270 terminal.

2.2.6 ► Bring up a 3270 terminal emulator clicking on the **host emulator icon** on the Windows task bar.



This opens the host emulator.

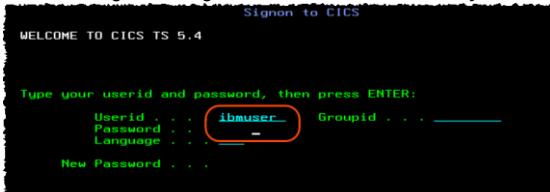
2.2.7 ► Type **I cicsts54.** (where I is L lower case) and press **Enter key**



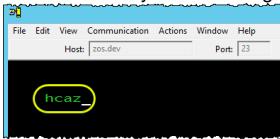
Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

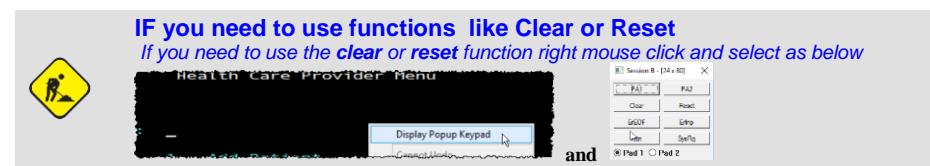
- 2.2.8 ►| Sign on using **ibmuser** as the userid and **sys1** as the password.



- 2.2.9 ►| Once you see the sign-on is complete message, enter **hcaz** and press the **Enter** key.



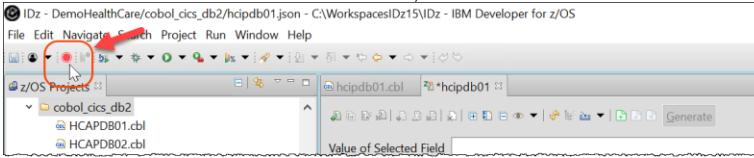
IF you need to use functions like Clear or Reset
If you need to use the clear or reset function right mouse click and select as below



You are now ready to start recording and import data into the test case editor.

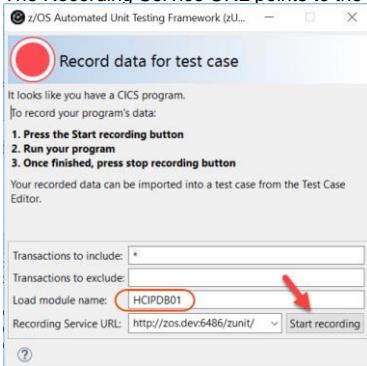
- 2.2.10 ►| Minimize the terminal emulator and go back to IDz dialog.

- | To record from a live run into the test case, click the **Record** button on the IDz toolbar.



- 2.2.11 ►| In the dialog that comes up, click on **Start recording**.

The Recording Service URL points to the CICS region where the live run is recorded.

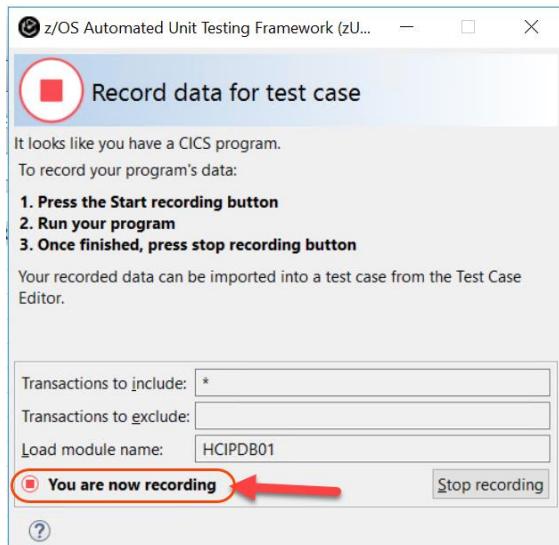


Exercises Guide

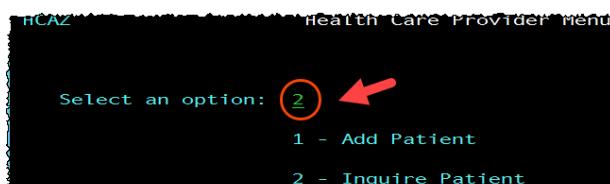
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

2.2.12 When recording is turned on, the message "You are now recording" appears.

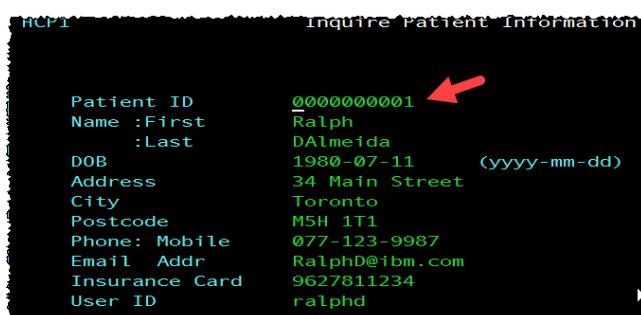
Verify that the Load module **HCIPDB01** is the one being recorded.

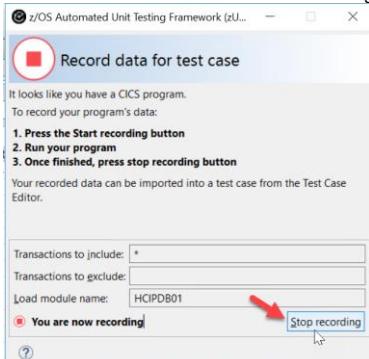


2.2.13 ► Switch to the 3270 terminal emulator (that is minimized), type **2** (Inquire Patient) and press **enter**. In case your transaction is not running just type **HCAZ** again.

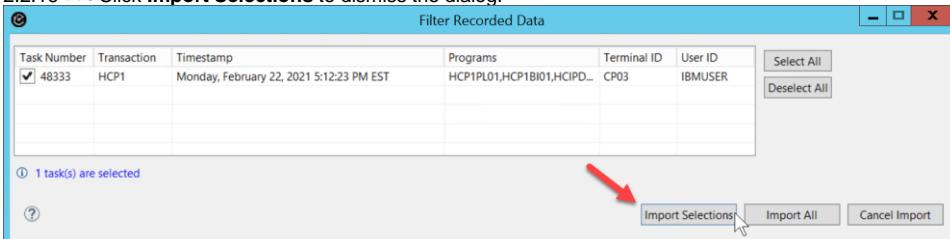
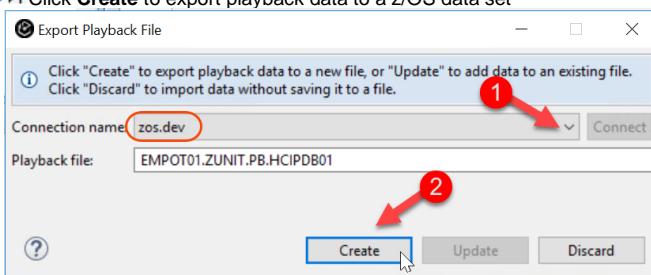
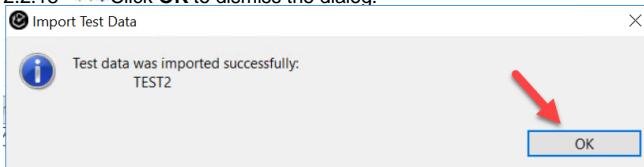


2.2.14 ► Type **1** and press **enter**



2.2.15 ►| Switch back to IDz minimizing the 3270 screen and click **Stop recording**.

The dialog *Filter Recorded Data* is displayed.

2.2.16 ►| Click **Import Selections** to dismiss the dialog.2.2.17 ►| Using drop down select the **zos.dev** as *Connection name*►| Click **Create** to export playback data to a z/OS data set2.2.18 ►| Click **OK** to dismiss the dialog.

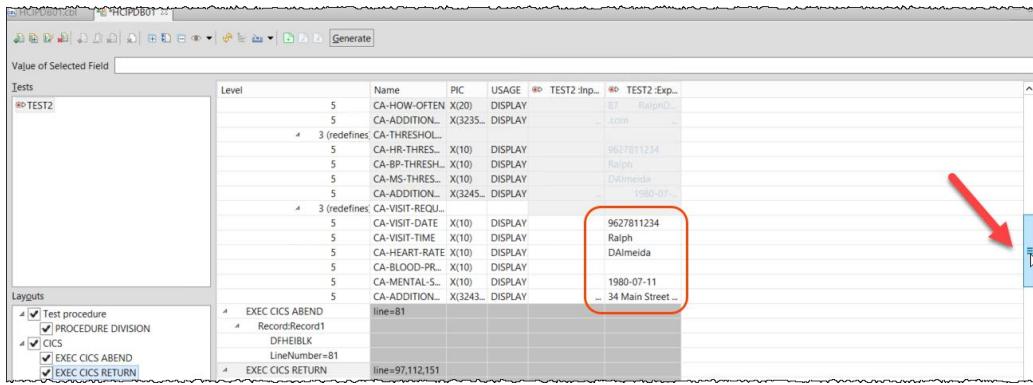
Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

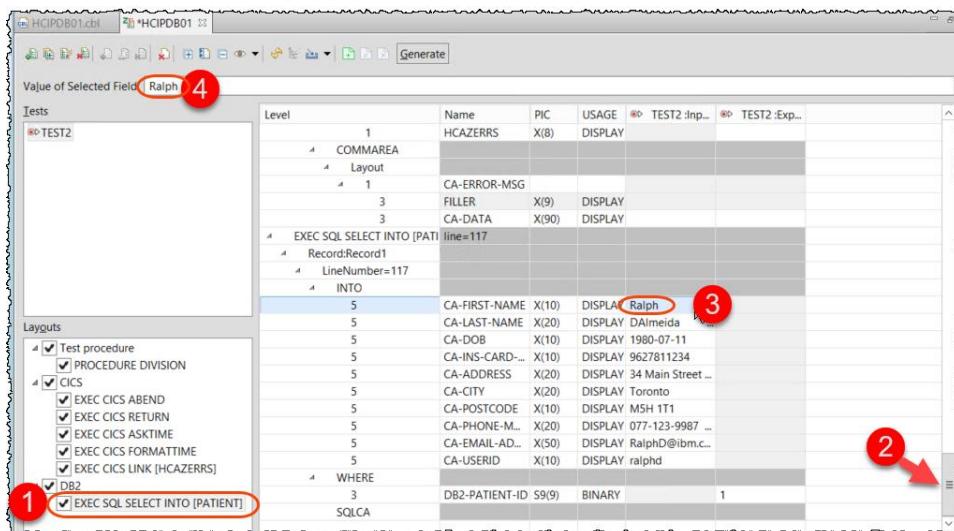
- 2.2.19 Double click on the **hcipdb01** title to enlarge the view.
In case by mistake you close this editor will need to go back to step 2.2.1.



- 2.2.20 You now see a new test created in the editor and populated with the values from the live run.
 **Scroll down** the editor and notice the data displayed on the screen that was captured..



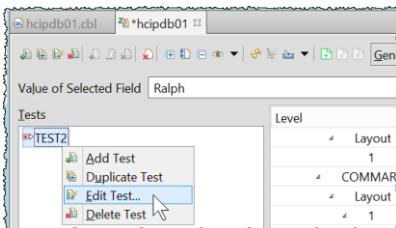
- 2.2.21 ► ① Click on **EXEC SQL SELECT INTO (PATIENT)** to position the data layout for this statement.
② Use the scroll bar to **scroll down** until the end,
③ Click on **Ralph** . and notice that value is displayed on the line ④ "



Exercises Guide

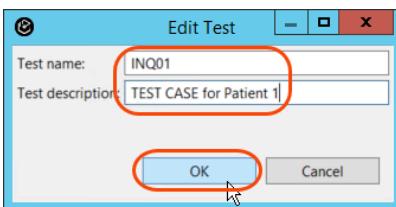
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

2.2.23 ► Right click on TEST2 and select Edit Test



2.2.24 It is a good practice give a name for the test case.

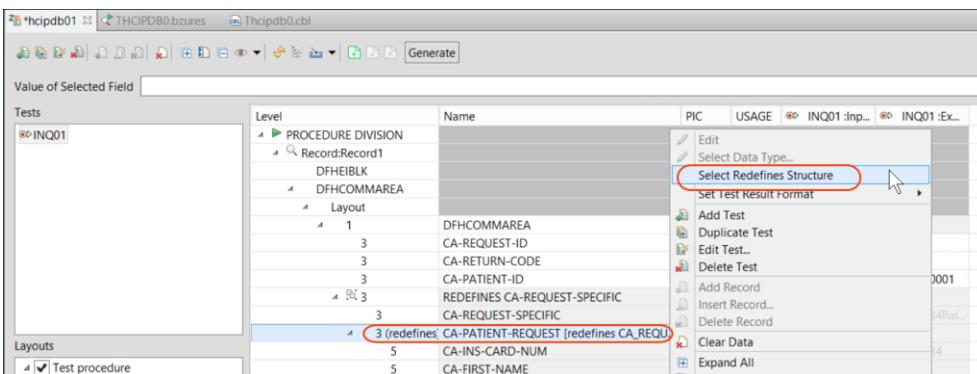
► Use a name like INQ01 and a description like "TEST CASE for Patient 1": . Click OK



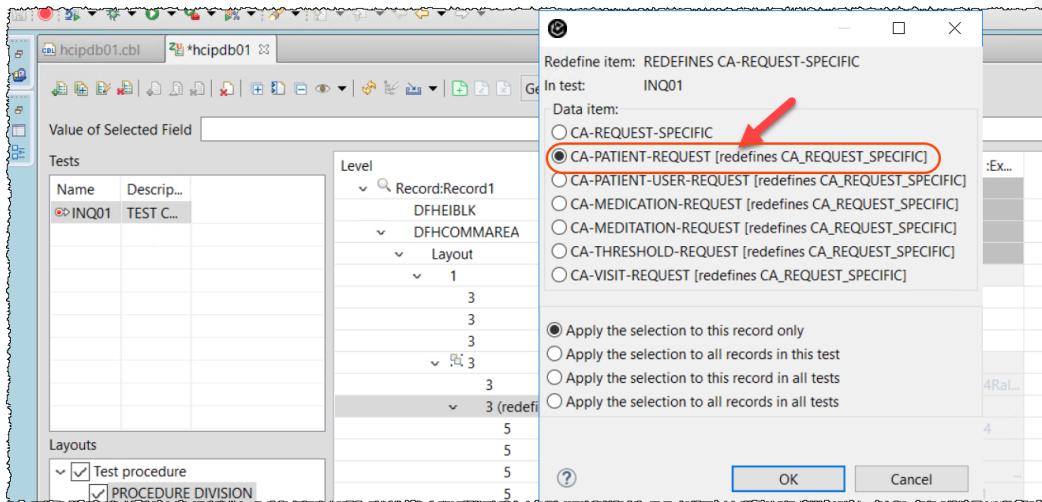
2.2.25 You may notice that this program has many redefines. You may need to identify which is the area being used on this program execution. In our example is the "inquiry patient" area being redefined.

► Scroll up until you find "(redefines) CA-PATIENT-REQUEST" (the first REDEFINES).

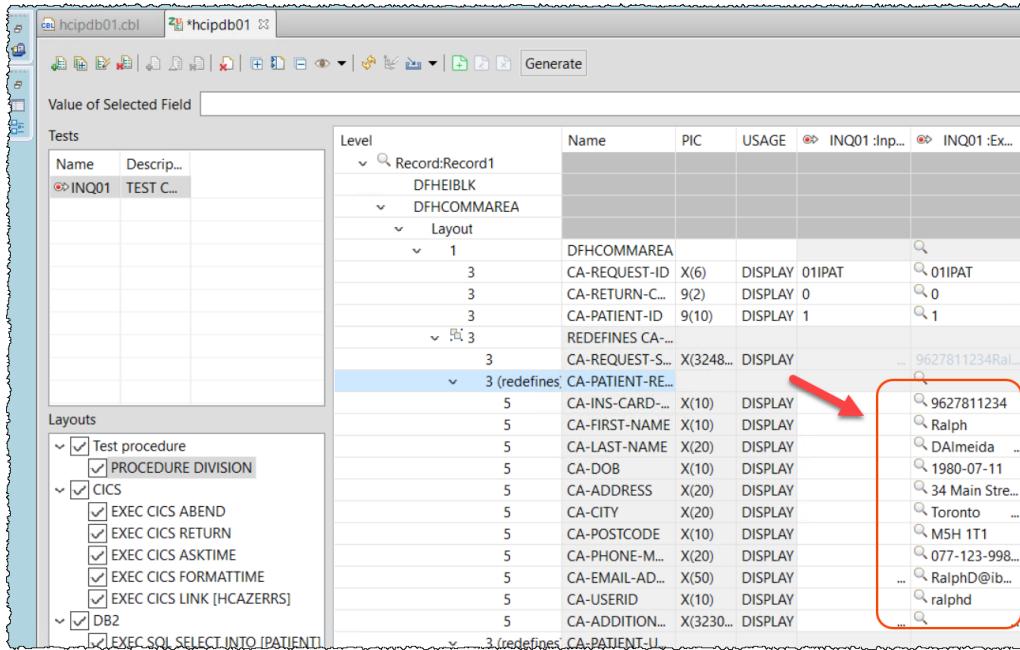
Right click on it and click Select Redefines Structure



2.2.26 Select CA-PATIENT-REQUEST and click OK



2.2.27 Notice that now the patient data recorded is displayed at the new redefines selected.



2.2.28 ► Press **Ctrl+S** to save any changes.

2.2.29 ► Double click again on the **hcipdb01** title to shrink the view.

In case by mistake you close this editor will can re-execute the step 2.2.1

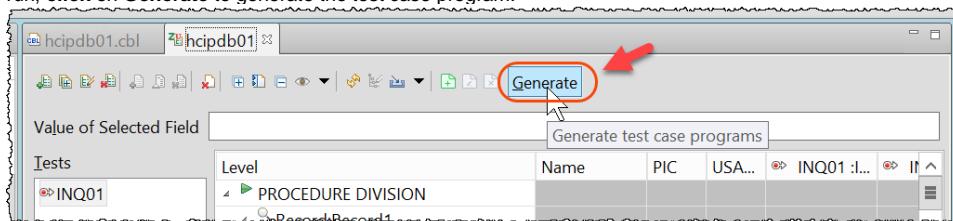


Section 3. Generate, build, and run the unit test.

You will generate, build, and run the unit test for the test case created.

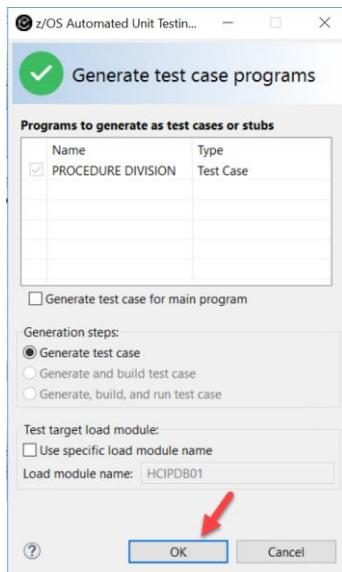
3.1 Generating the test case program.

3.1.1 ► Now that the expected input and output values have been set in the test case editor from the recorded run, click on **Generate** to generate the test case program.



On the **Generate test case programs** dialog,

3.1.2 ► Click **OK**.



Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.1.3 The COBOL program that will run the test case is generated under the folder *testcases*.

► Expand **testcases** and double click on **Hcipdb01.cbl** to verify the generated program.

Notice that in fact **9 COBOL programs** were generated from this test case . This can be seen on the **Outline** view. You could navigate to each program if time allows, note that **NONE** of these programs will require CICS or DB2 to be executed. All will be executed in batch using JCL.

The screenshot shows the Rational Developer for z/OS interface. In the center, the code editor displays the COBOL source code for **Hcipdb01.cbl**. The code includes comments indicating it's a test case for dynamic runner, generated on 03/12/2021 at 11:45, with an ID of f5efa266-00b7-4a39-a2f9-81d240238bd9. It defines a TEST_INQ01 program. In the left pane, the **z/OS Projects** view shows the project structure with **testcases** expanded, and **Hcipdb01.cbl** highlighted with a red circle and arrow. The **Outline** view in the bottom left shows a list of generated programs like **TEST_INQ01**, **BZU_TEST**, etc.

```
1  * PROCESS NOJLL,NOJNAM,TEST(NOSEP),NOJCS,NOSQL,PGMN(LU)
2  *
3  *| Thcipdb0
4  *| PROJECT: IBM DEVELOPER FOR Z/OS
5  *| COMPONENT: IBM Z/OS AUTOMATED UNIT TESTING FRAMEWORK (ZUNIT)
6  *| FOO ENTERPRISE COBOL AND PL/I
7  *| PROGRAM: ENTERPRISE COBOL UNIT TEST CASE FOR DYNAMIC RUNNER
8  *| DATE GENERATED: 03/12/2021 11:45
9  *| ID: f5efa266-00b7-4a39-a2f9-81d240238bd9
10 *
11 *| TEST_INQ01
12 *| THIS PROGRAM IS FOR TEST INQ01
13 *
14 IDENTIFICATION DIVISION.
15* PROGRAM-ID. 'TEST_INQ01'.
16*
17* DATA DIVISION.
18* WORKING-STORAGE SECTION.
19* 01 PROGRAM-NAME PIC X(8) VALUE 'HCIPDB01'.
20* 01 BZ-ASSERT.
21* 03 MESSAGE-LEN PIC S9(4) COMP-4 VALUE 24.
```

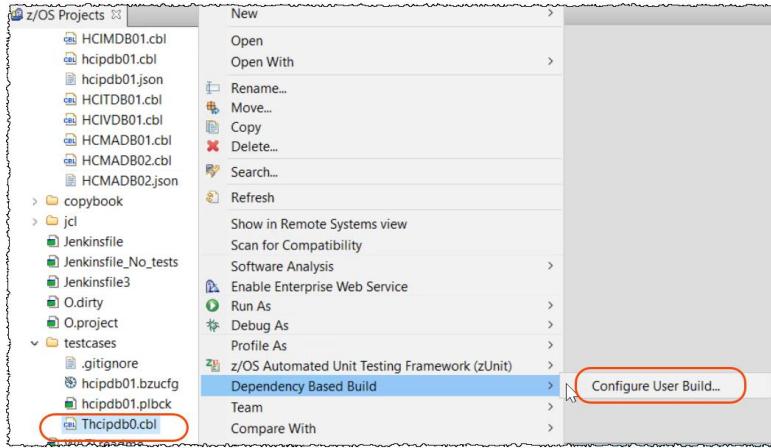
3.2 Building the generated test case programs using IBM DBB .

The 6 generated COBOL programs need to be compiled/link edited to be executed. This must be done on z/OS using the COBOL compiler.

To make this easier we will use the IBM DBB. We provided the required framework to make this possible. But note that this could be done using any other way, including via JCL normal compilation.

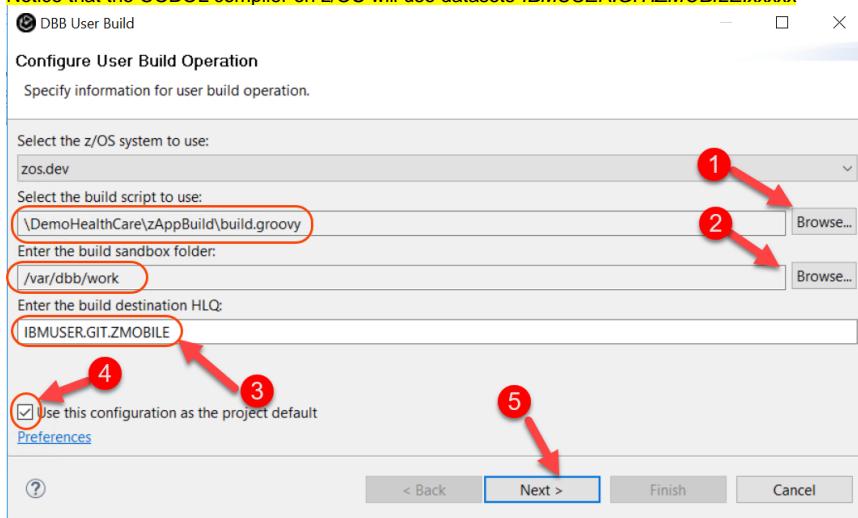
3.2.1 ► Use **Ctrl + Shift + F4** to close all opened editors.

3.2.2 ► Right click on **Hcipdb01.cbl** (under *testcases*) and select **Dependency Based Build > Configure User Build**.



3.2.3 ► Those values could be already there, otherwise use the **Browse...** buttons to specify the values as below. As build destination HLQ use **IBMUSER.GIT.ZMOBILE**, select **Use this configuration as the project default** and click **Next**

Notice that the COBOL compiler on z/OS will use datasets **IBMUSER.GIT.ZMOBILE.xxxx**



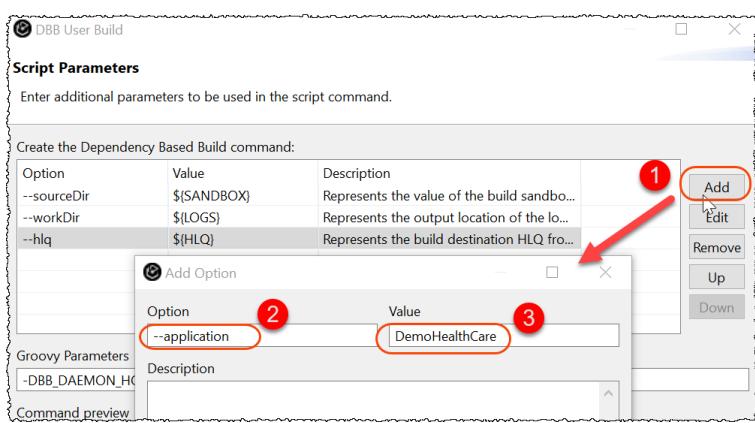
3.2.4 ► Click Next Twice

3.2.5 On dialog **Script Parameters**, if the “**--application**” and “**--debug**” are not there you must add

► Otherwise skip to 3.2.7

In case you need to add **--application**.

► Click **Add** to insert **--application** and **DemoHealthCare** (mixed case) and click **OK**



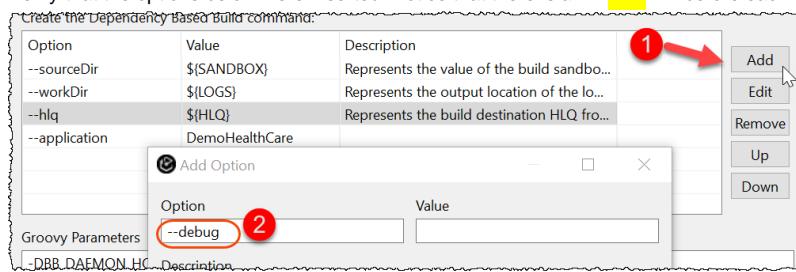
Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.2.6 In case you need to add --debug.

- ▶ Click Add to insert `--debug` and click OK

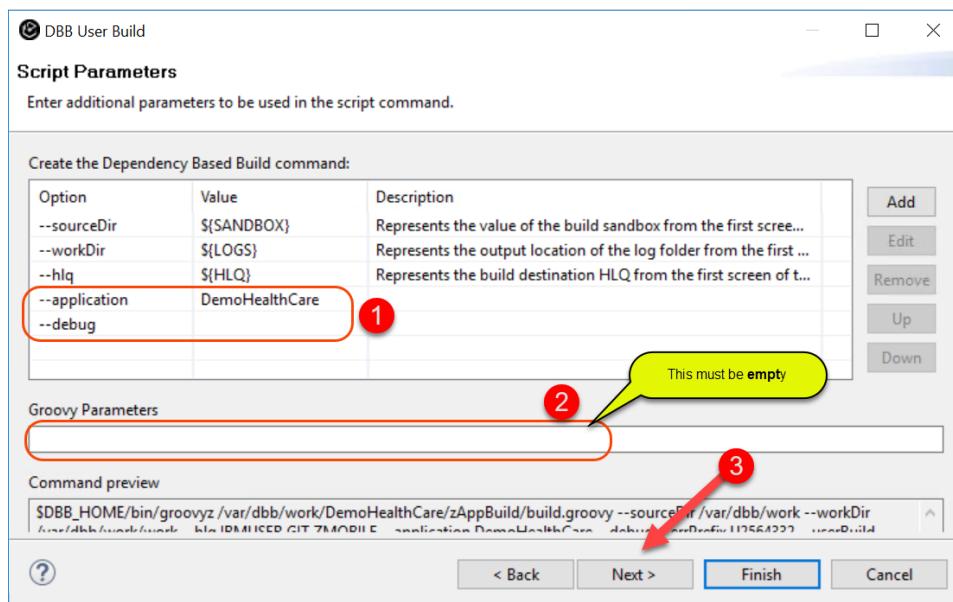
Verify that the options below were inserted. Notice that there is an “`--`” before each option.



3.2.7 The field Groovy Parameters should be cleared if there is something already specified there.

- ▶ Be sure that the Groovy parameters field is empty

- ▶ Click Next

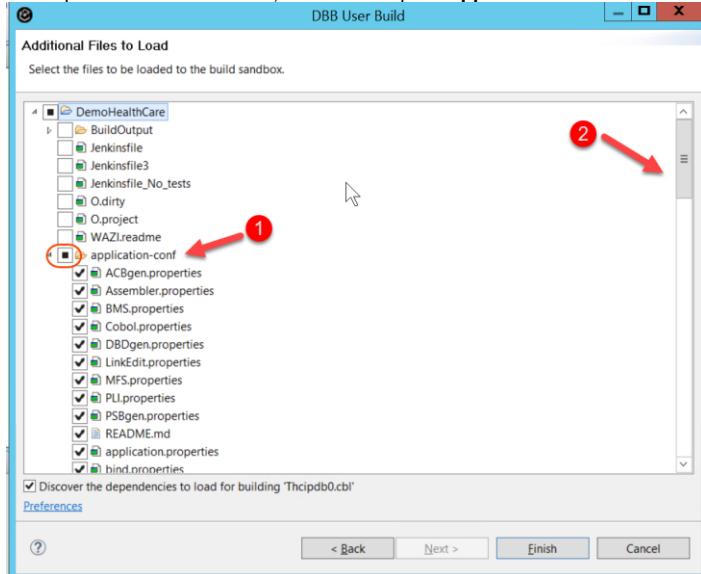


Exercises Guide

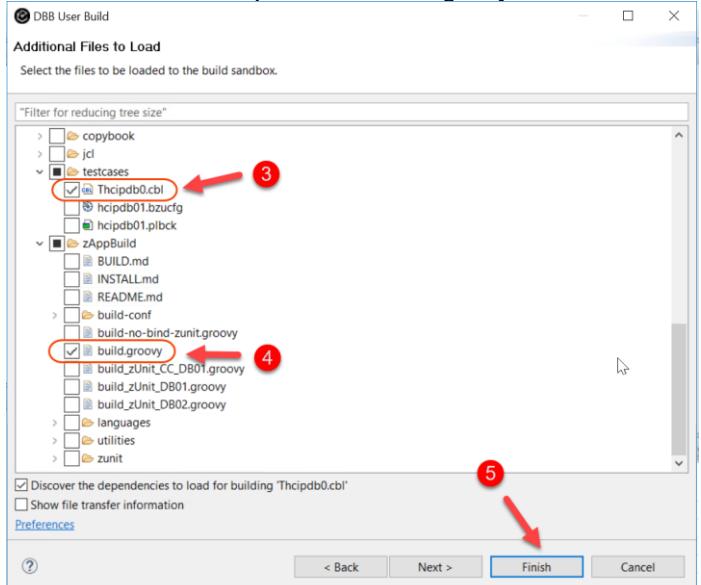
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.2.8 You need to move the assets to z/OS UNIX Files

► Expand DemoHealthCare, select and expand application -conf and scroll down

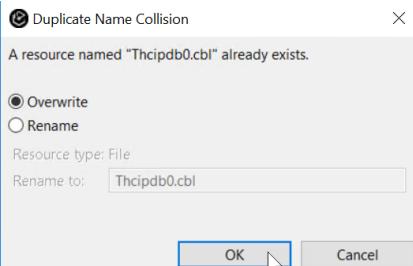


3.2.9 ► Be sure that Thcipdb0.cbl and build.groovy are selected and click Finish

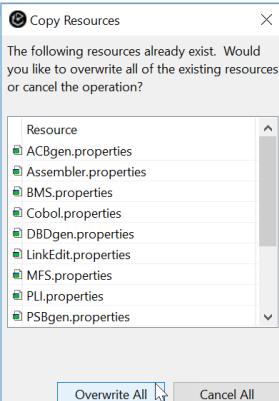


3.2.9 Those assets were once already moved to the z/OS UNIX files and will overwrite the old copies.

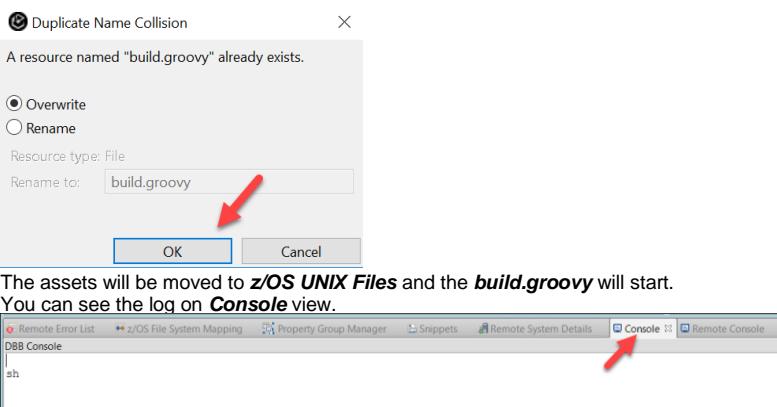
► Click OK



▶ Click Overwrite All



▶ For duplicate name collision click OK



If the DBB user build fails
If you have errors as below: (Resources temporary unavailable)

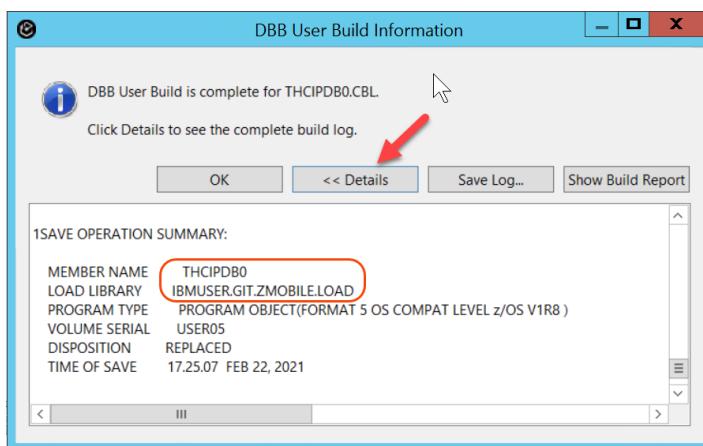


DBB Console

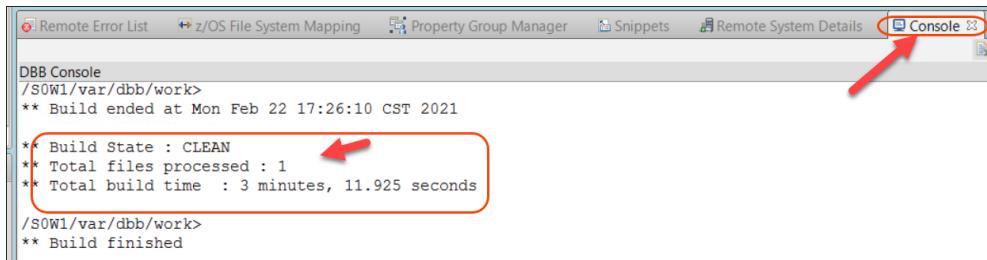
```
Cannot run program "sh" (in directory "/var/dbb/work"): EDC5112I Resource temporarily unavailable
$DBB_HOME/bin/groovy: -DBB_DAEMON_HOST 127.0.0.1 -DBB_DAEMON_PORT 8080 /var/dbb/work/DemoHealthCare/zAppBuild/build.groovy --s
It's because your z/OS is stressed,, One easy way to free the z/OS resources is logging on TSO as
IBMUSER password SYS1, go to M.5 and issue /C CICSTS53 And Try again
```

3.2.10 This operation will compile, and link the 9 generated COBOL programs **and it may take up to 3 minutes.**
 You can follow this using the **Console view** opened in the bottom.

► When finished the dialog below will be displayed and you can click on **Details** and **OK** after you scroll down and verify that the load module was created.



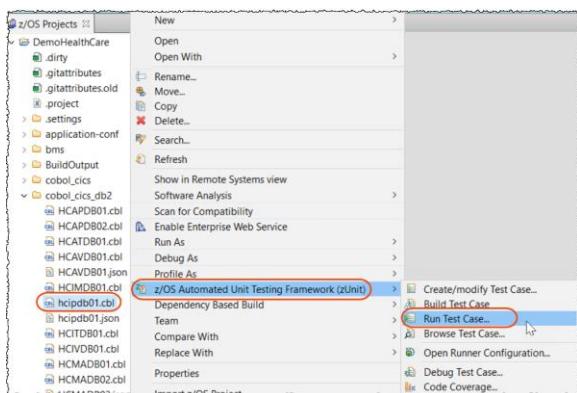
► The **Console view** also show the results as below:



3.3 Running the test case,

Once you have the test cases load module created you can run the test case against the COBOL program. Since you did not make changes to the program the return code must be zero and all tests should pass.

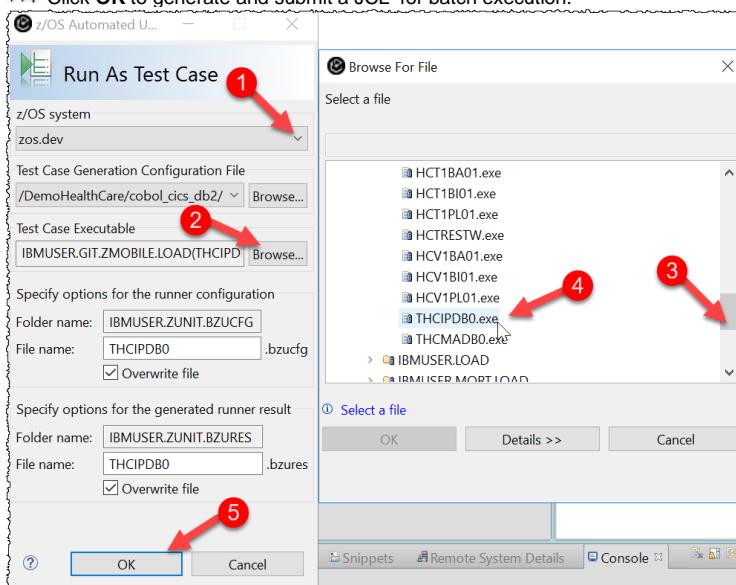
- 3.3.1 ► Using z/OS Projects view, scroll up, right click on **hcipdb01.cbl** and select **z/OS Automated Unit Testing Framework (zUnit)->Run Test Case...**



- 3.3.2 ► Select **zos.dev** and use the **Browse** button to select the PDS and load module where the test case was created.

You will find the PDS **IBMUSER.GIT.ZMOBILE.LOAD(THCIPDB0)** under **My Data Sets ..**

- Click **OK** to generate and submit a JCL for batch execution.



Exercises Guide

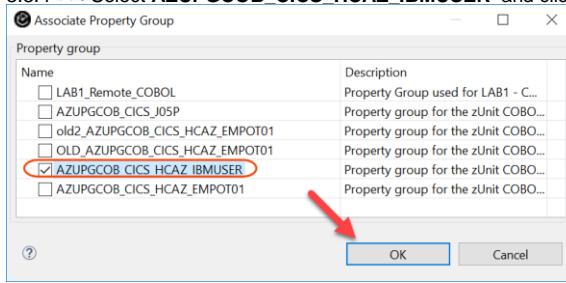
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.3.3 If the Missing Property Group dialog will prompt.

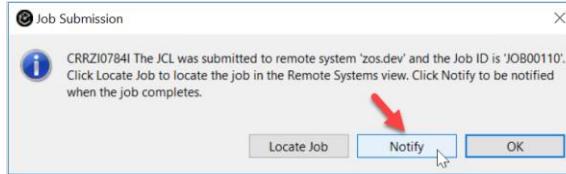
► Click **Associate an existing property group** otherwise go to 3.3.5



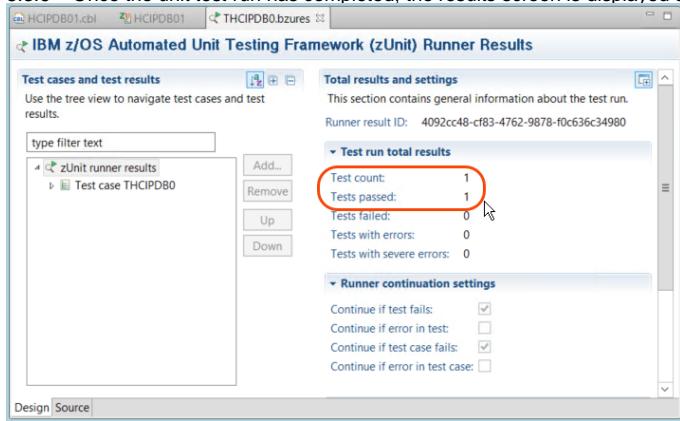
3.3.4 ► Select **AZUPGCOB_CICS_HCAZ_IBMUSER** and click **OK**



3.3.5 ► A Job Submission dialog opens, click **Notify** to be notified of job completion.



3.3.6 Once the unit test run has completed, the results screen is displayed showing test cases ran and passed.



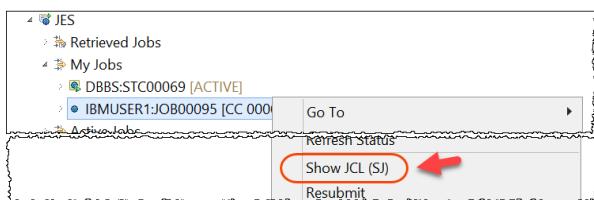
You have now created a test case with data imported from a recorded run and have been successful in testing a CICS program without the need of a CICS and DB2 environment.

3.3.7 ►| Use **Ctrl + Shift + F4** to close all opened editors.

3.4 Verify the JCL submitted

To see the JCL submitted on the previous execution

3.4.1 ►| Using **Remote Systems** view on right, under **JES** expand **My Jobs**, right click on the last executed that has **CC 000** (NOT the ACTIVE) and select **Show JCL (SJ)**



3.4.2 ►| The job submitted will be displayed. You could save it in a PDS member and use it when want to run the test cases for this program.

As you see there is no CICS or DB2 environments in that execution.

3.4.3 ►| Use **Ctrl + Shift + F4** to close all opened editors.

3.5 Running zUnit test case Code Coverage

Code coverage analyzes a running program and generates a report of lines that are executed, compared to the total number of executable lines. Sometimes you need to be sure that your test is covering all capabilities of your program and that you are having the correct test cases.

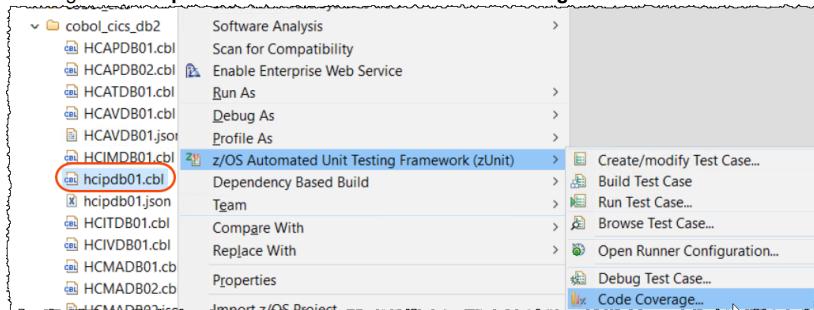
Notice that you are doing the code coverage using a batch job without need to have CICS and DB2 active, which is very handy..

Exercises Guide

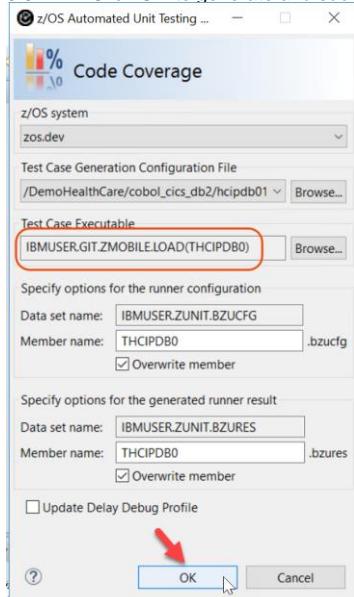
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.5.1 ZUnit also provides the capability to run the test case using Code coverage or Debug.

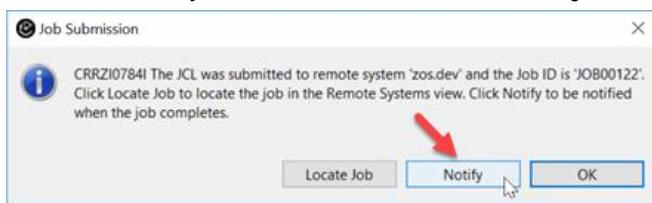
► Right click **hcipdb01.cbl** and select **zUnit > Code Coverage....**



3.5.2 ► Click **OK** to generate and submit a JCL for batch execution.



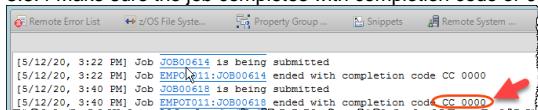
3.5.3 ► Click **Notify** on the Job Submission Confirmation dialog.



Exercises Guide

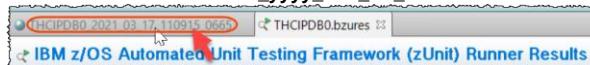
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

3.5.4 Make sure the job completes with completion code of 0000.



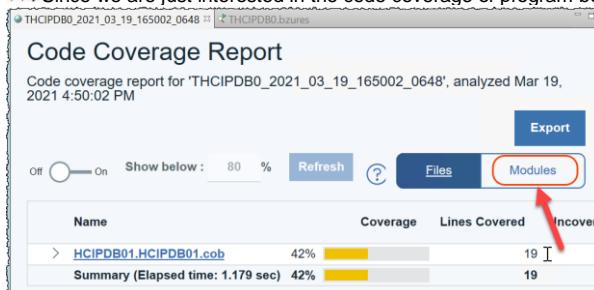
3.5.5 The code coverage report shows all COBOL programs executed for this specific test case.

▶ Click on title **THCIPDB0_yyyy_mm_dd_xxxxxxx** to see the Code coverage report



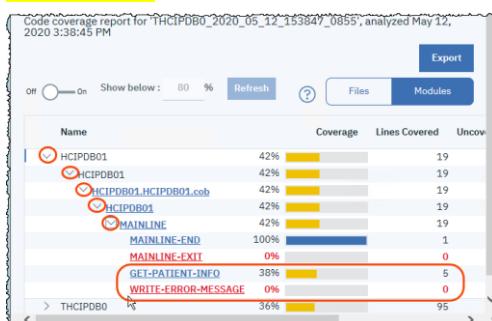
3.5.6 This code coverage report shows the coverage of the COBOL programs generated for zUnit to perform the test as well our **HCIPDB01** COBOL program..

▶ Since we are just interested in the code coverage of program being tested click on **Modules**:



3.5.7 ▶ Expand the nodes as below to see the COBOL paragraphs.

Notice that **GET-PATIENT-INFO** has 38% of coverage and that **WRITE-ERROR-MESSAGE** was not covered at all on this test case.



3.5.8 ▶ Click on **GET-PATIENT-INFO** to see more details



3.5.9 Scroll down and move the mouse to the colored areas on left.

The lines covered by this test case are in **green** and the lines not covered are in **red**.

Also from the previous image we can see the WRITE-ERROR-MESSAGE paragraph is not covered at all.
(you may have different values than the screen capture below).

```

JOB00096.jcl          THCPDB0_20...   THCPDB0.bzures   HCIPDB01.HC... >
-----+-----+-----+-----+-----+-----+-----+
255 :CA-ADDRESS,
256 :CA-CITY,
257 :CA-POSTCODE,
258 :CA-PHONE-MOBILE,
259 :CA-EMAIL-ADDRESS,
260 :CA-USERID
261 FROM PATIENT
262 WHERE PATIENTID = :DB2-PATIENT-ID
263 END-EXEC.
264 Evaluate SQLCODE
265 When 0
266 MOVE '00' TO CA-RETURN-CODE
267 Lines 267-274 not covered. 00
268           MOVE '01' TO CA-RETURN-CODE
269 When -913
270 MOVE '01' TO CA-RETURN-CODE
271 When Other
272 MOVE '90' TO CA-RETURN-CODE
273 PERFORM WRITE-ERROR-MESSAGE
274 EXEC CICS RETURN END-EXEC
275
276 END-Evaluate

```

3.5.10 From this report we can see that the test cases created for this program are not sufficient ..

The developer could go back to the test cases editor (step 2.2.20 above) and manually add test cases that cover other paragraphs. Due to the lack of time we will not cover that here, but if you have extra time you can try it. Ask the instructor for guidance.

3.5.11 We also saved the JCL showed on step 3.4.2 and added the information to run the code coverage.

This JCL can be found at **jcl/Hcipcc.jcl**

Double click on **jcl/Hcipcc.jcl** and verify this JCL. You could just submit this JCL to run the test case and the Code Coverage.

```

z/OS Projects  THCPDB0_20...   THCPDB0.bzures   HCIPDB01.HC... >
-----+-----+-----+-----+-----+-----+-----+
copybook
  jcl
    HCZPLA2.jcl
    HCZPLA3.jcl
    HCZPLAY.jcl
    HCZPLCC.jcl
    HCIPCC.jcl
    HCIPDBG.jcl
    HCIPZRUN.jcl
    VTP#HCAZ.jcl
    VTP#TES1.jcl
    VTP#TES2.jcl
    ZUDBB01.jcl
    ZUDBB02.jcl
  HCIPCC JOB ,
  // MSGCLASS=H, MSGLEVEL=(1,1), REGION=0M, COND=(16,LT)
  // Action: Code Coverage...
  // Source: IBMUSER.GIT.ZMOBILE.LOAD(THCPDB0)
  // RUNNER EXEC PROC=BZUPPLAY,
  // BZUCFG=IBMUSER.ZUNIT.BZUCFG(THCPDB0),
  // BZUCBK=IBMUSER.ZUNIT.BZUCBK(THCPDB0),
  // BZULOD=IBMUSER.GIT.ZMOBILE.LOAD,
  // PARM='('STOP=E,REPORT=XML')'
  // CEEOPTS DD *
  TEST(ALL,,PROMPT,DEMDT%IBMUSER:*)
  ENVAR(
  "EQA_STARTUP_KEY=CC,THCPDB0,cclevel=LINE,testid=THCPDB0")
  /*
  BZUPLAY DD DISP=SHR,
  DSN=IBMUSER.ZUNIT.PB1.HCIPDB01
  BZURPI DD DISP=SHR,
  DSN=IBMUSER.ZUNIT.BZURES(THCPDB0)

```

Section 4. Introduce a bug in the program and rerun the unit test.

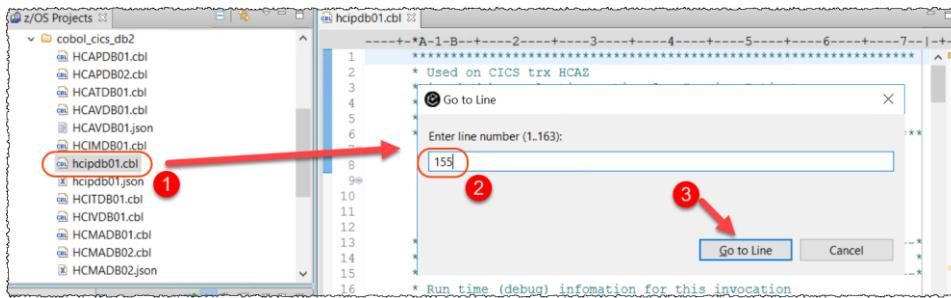
Using IDz you will modify the program and introduce a bug. Then you will rerun the unit test created in the previous section.

4.1 Modifying the program and introduce a bug

4.1.1 ► On IDz, close all active windows by pressing **Ctrl+Shift+F4**

► Open **hcipdb01.cbl** under **cobol_cics_db2** by double clicking on it in the z/OS Projects view.

4.1.2 ► On the editor, go to line 155 by pressing **Ctrl+L** and entering **155** and clicking **Go to Line**



4.1.3 ► Change the lines 156, 157 and 158 removing the * from the statement that moves "BAD NAME",
Tip -> Could use **Source > Toggle Comment** also **DO NOT** change line 159.

► Press **Ctrl+S** to save the changes. and **Ctrl + Shift + F4** to close all editors.

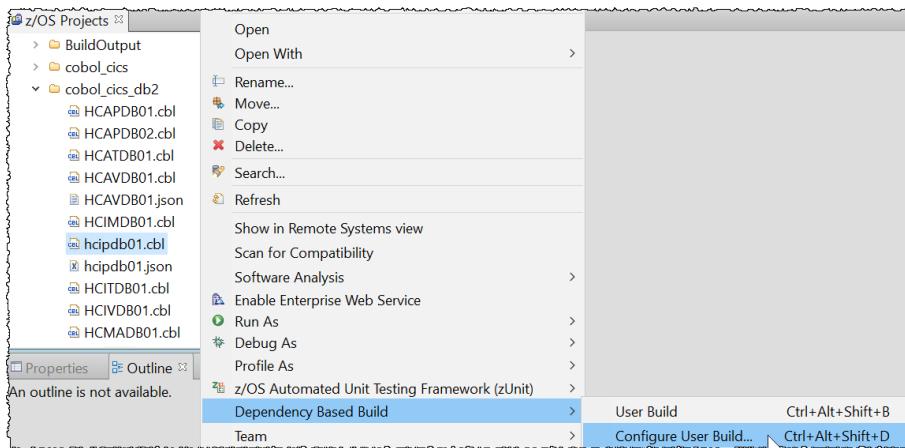
```

141      END-EXEC.
142      Evaluate SQLCODE
143      When 0
144          MOVE '00' TO CA-RETURN-CODE
145      When 100
146          MOVE '01' TO CA-RETURN-CODE
147      When -913
148          MOVE '01' TO CA-RETURN-CODE
149      When Other
150          MOVE '90' TO CA-RETURN-CODE
151          PERFORM WRITE-ERROR-MESSAGE
152          EXEC CICS RETURN END-EXEC
153      END-Evaluate.
154      *%bug2 -- the line below will introduce a BUG
155      *
156      IF DB2-PATIENT-ID = 1
157          MOVE "BAD NAME" to CA-FIRST-NAME
158      END-IF
159      *
160      MOVE "02" to CA-NEWFIELD
161      *
162      EXIT.
163      *
164      COPY HCERRSPD.

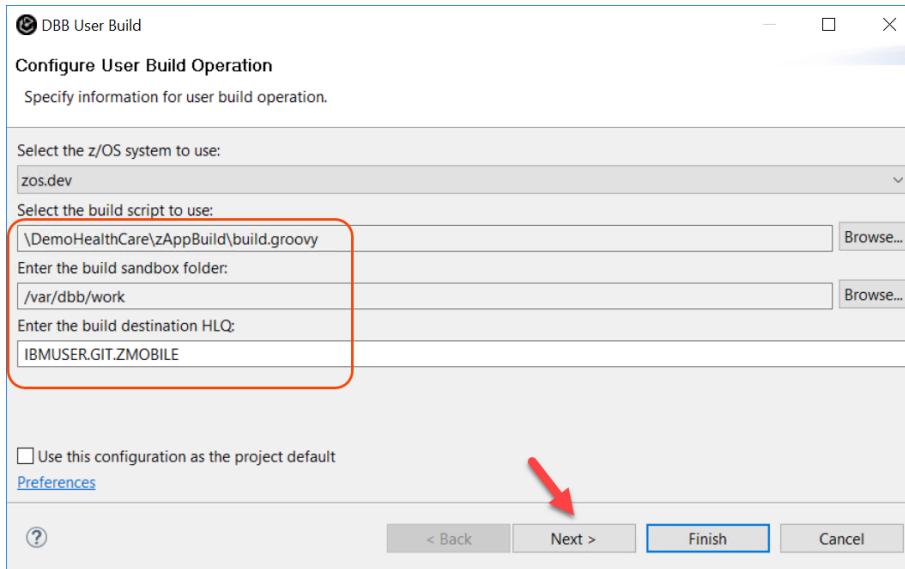
```

4.2 Rebuilding the changed program without deploying to CICS using DBB

- .4.2.1 ► On the z/OS Projects view, right click on **hcipdb01.cbl** and select **Dependency Based Build > Configure User Build...**



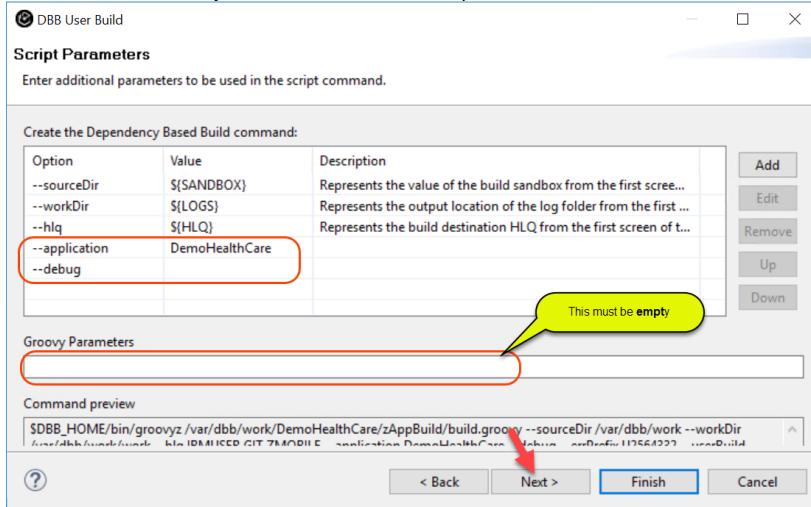
- .4.2.2 ► If the values are not populated, use the Browse button and specify the values below and click **Next 3 times**



Exercises Guide

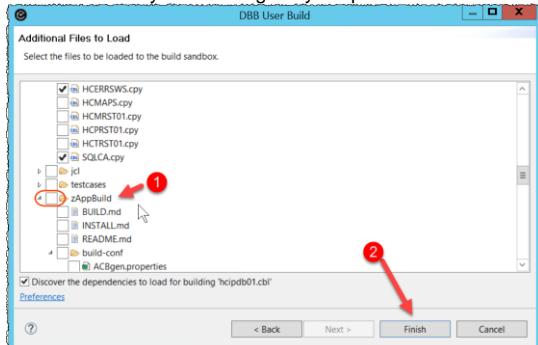
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.2.3 ► Be sure that you have the values below specified and click **Next**

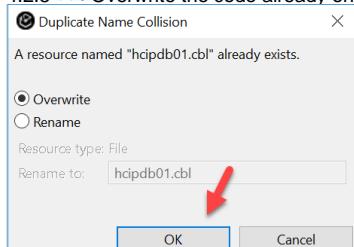


4.2.4 ► Expand DemoHealthCare, Un-select zAppBuild and click **Finish**.

You have already moved all groovy scripts to z/OS before when compiled the Test case program.



4.2.5 ► Overwrite the code already on z/OS as the example below

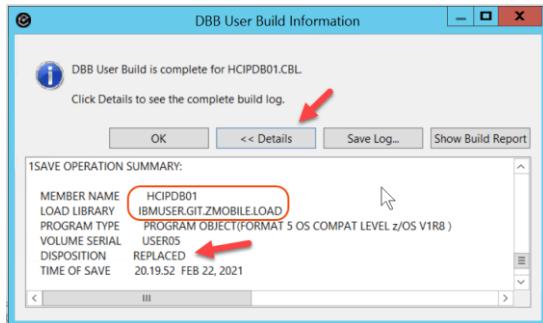


Exercises Guide

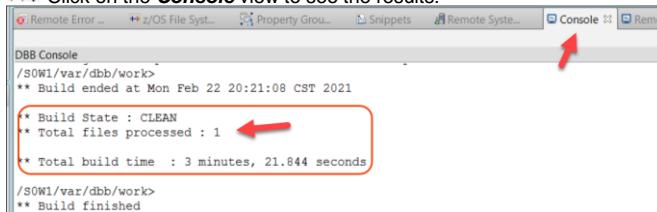
Materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.2.6 This operation may take up to 2 minutes When finished the dialog below will be displayed.

► Click on **Details** and **OK** after you verified that the load module was replaced.



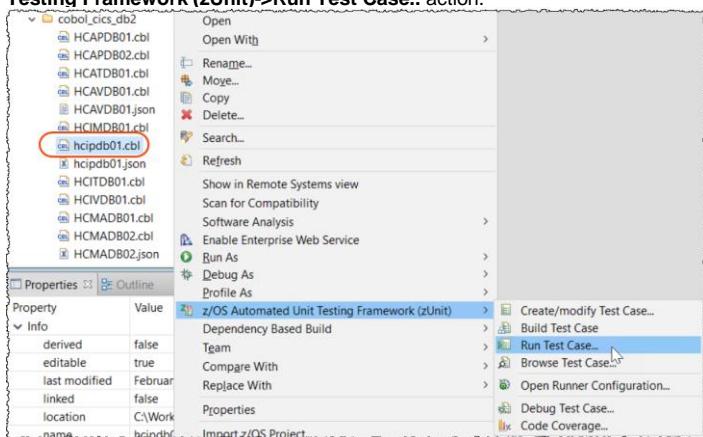
► Click on the **Console** view to see the results:



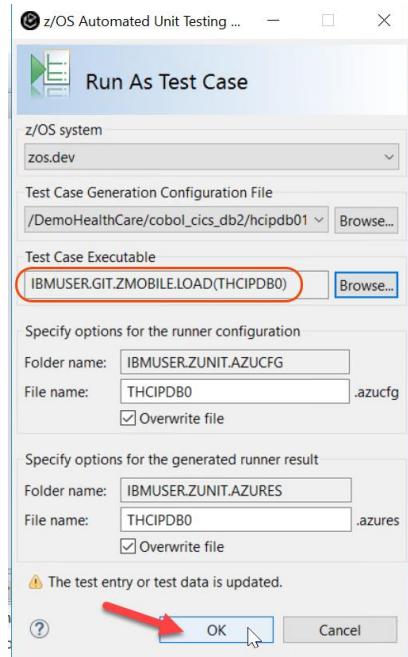
4.3 Running the test case again

Since we introduced a bug, now the test case must fail.

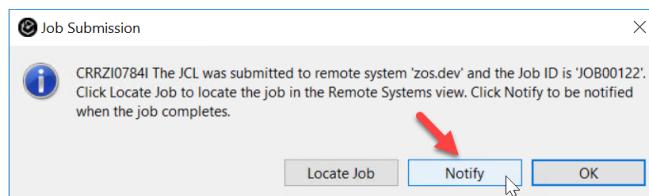
4.3.1 ► On the z/OS Projects view, select **hcipdb01.cbl**, right mouse click, and select the **z/OS Automated Unit Testing Framework (zUnit)**->**Run Test Case..** action.



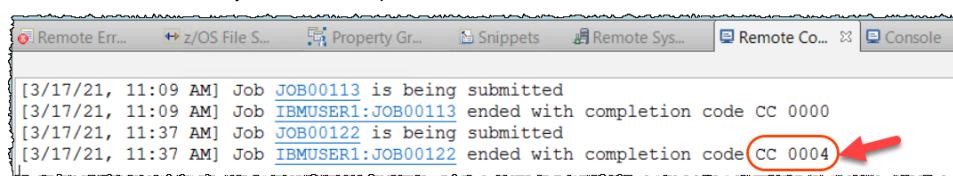
4.3.2 ► Be sure that the test case load module **IBMUSER.GIT.ZMOBILE.LOAD(THCIPDB0)** is selected and click **OK** to submit the execution via JCL.



4.3.3 ► Click **Notify** on the Job Submission Confirmation dialog.



4.3.4 ► Notice that now you have a completion code of **0004** instead of 0000.



Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.3.5 When the test run completes, the test results will be displayed, and it showed 1 test was ran and it failed.

► Click **Test case THCIPDB0**. The failure is expected because the expected value of the error message is different from the actual value.

IBM z/OS Automated Unit Testing Framework (zUnit) Runner Results

Test cases and test results

Test case details

Test case ID: e30e513c-140b-4176-b4f1-0b497de49f44
Module name: THCIPDB0
Test case name: THCIPDB0
Result: fail

| | |
|---------------------------|---|
| Test count: | 1 |
| Tests passed: | 0 |
| Tests failed: | 1 |
| Tests with errors: | 0 |
| Tests with severe errors: | 0 |

4.3.6 ► Double click on title **THCIPDB0.bzures** to maximize the windows and better check the results

THCIPDB0.bzures

IBM z/OS

4.3.7 ► Expand **Test case THCIPDB0, Test INQ01 (fail)** and click **Exception message number** to verify the value received versus the expected value and verify the failure

Exception details

If the result of the test is "fail" or "error", this section contains the details of the exception.

```
BZUPU00W ASSERT=COMPARE FAILED IN PROCEDURE DIVISION.  
BZUP220I TEST RUN TEST2 REGISTERED FOR SUBSYS=CICS FILTER ON=HCIPDB01 STUB CALL=Y  
BZUP400I STARTING TRANSACTION=HCP1 USING PROGRAM=TEST2  
BZUPU00W ASSERT=COMPARE FAILED IN PROCEDURE DIVISION.  
BZUPT00I ITEM NAME=CA-VISIT-TIME OF CA-VISIT-REQUEST OF DFHCOMMAREA  
BZUPT00I VALUE=BAD NAME  
BZUPT00I EXPECTED VALUE=Ralph  
BZUP002I FINISHED EXECUTIC RC=04
```

Message:

Component code: BZU

Message number: 3000

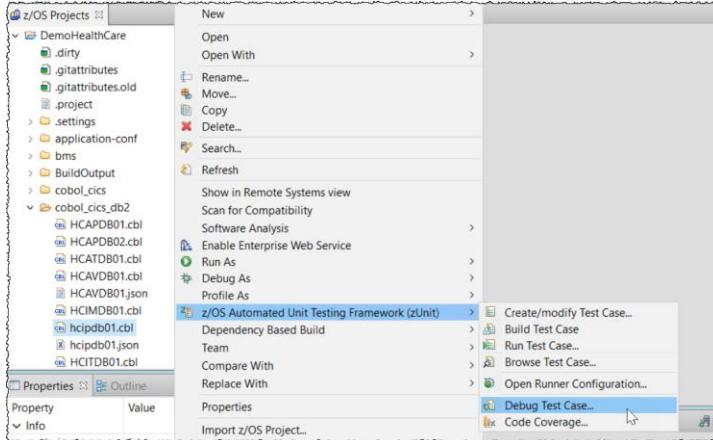
Message severity: 4

4.4 Running zUnit test case with debugging

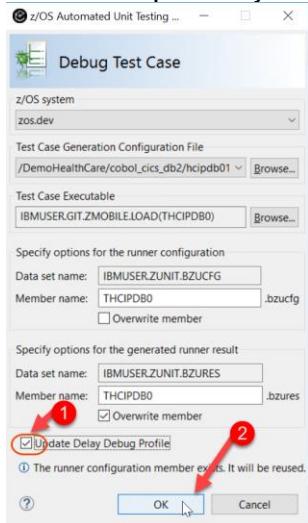
The Debug dialog will debug the test case. Note that will debug the zUnit programs, the generated COBOL and also the COBOL program that you are testing. Below one example. Notice that you are debugging a batch job without need to have CICS and DB2 active, which is very handy.

4.4.1 ► Close all active windows by pressing **Ctrl+Shift+F4**

4.4.2 ► Right click on **hcipdb01.cbl** and select **zUnit → Debug Test Case ...**



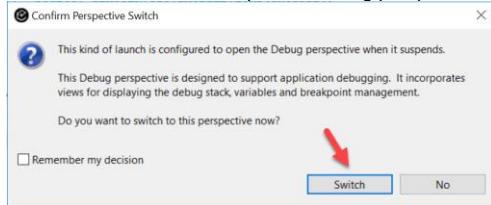
4.4.3 ► Click **Update Delay Debug Profile** and click **OK**



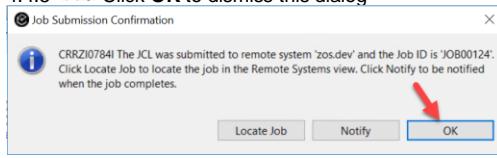
Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.4.4  Click **Switch** to open the debug perspective.



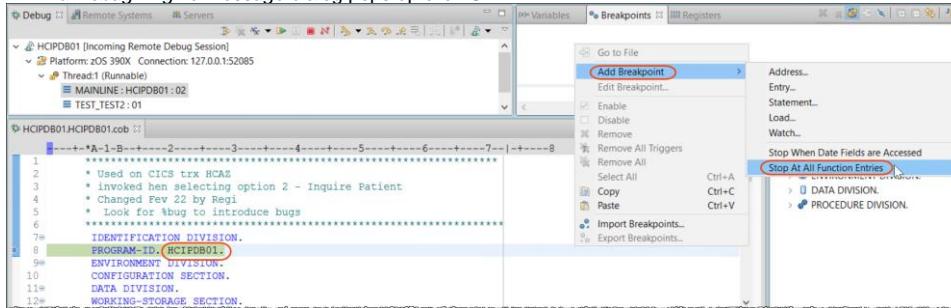
4.4.5 ► Click **OK** to dismiss this dialog



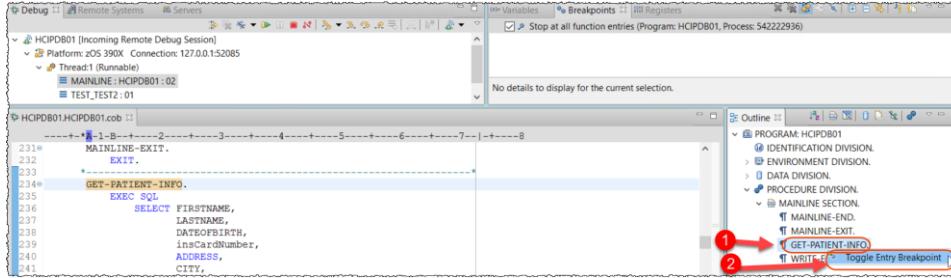
4.4.6 Notice that the first program being debugged is the program that you introduced the bug. (**HCPDB01**) The execution is stopped BEFORE that program starts. You can add some breakpoints before running.

Using the Breakpoints view, right click and select **Add Breakpoints > Stop At All Functions Entries**. This would allow you to reach the entry point of the source files by repeatedly clicking *resume*.

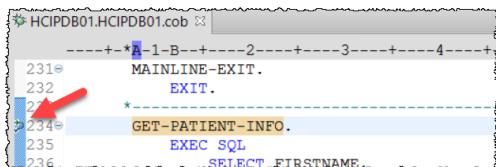
If a Debug Engine Message dialog pops up click **OK**



4.4.7 ► Using the **Outline** view to navigate to **GET-PATIENT-INFO**, right click and select **Toggle Entry Breakpoint**



4.4.8 A breakpoint is created on the **EXEC SQL SELECT** statement. When the program runs it will stop there.



```

HCIPDB01.HCIPDB01.cob:3
-----+---1-B---+---2---+---3---+---4---+
      MAINLINE-EXIT.
231*
232*
233*
234* GET-PATIENT-INFO.
235   EXEC SQL
236     SELECT FIRSTNAME
237
238
239
240
241
242

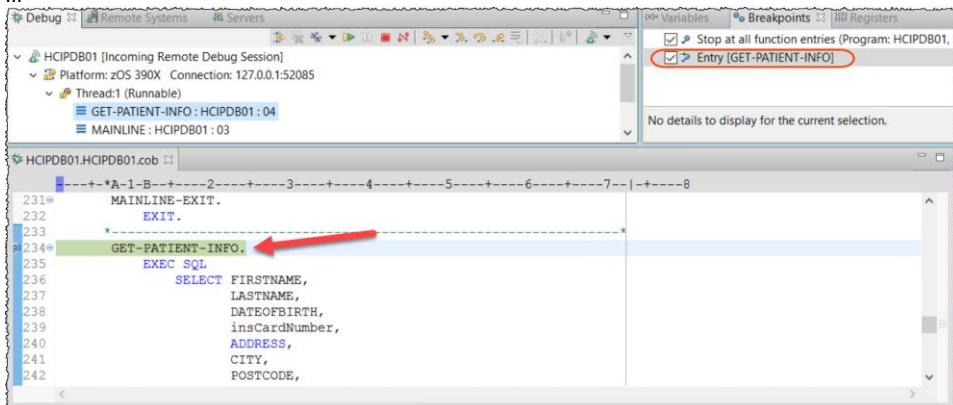
```

4.4.9  Click on icon  or press F8 to resume the execution



4.4.10 The execution will halt at the **SQL SELECT** statement due the breakpoint you added.

Notice your program is now using "stubbed code data" instead of accessing the DB2 database..



The Breakpoints view shows a list of breakpoints:

- Stop at all function entries (Program: HCIPDB01, P)
- Entry [GET-PATIENT-INFO]

The code editor shows the COBOL source code with line 234 highlighted:

```

HCIPDB01.HCIPDB01.cob:3
-----+---1-B---+---2---+---3---+---4---+---5---+---6---+---7---+---8
      MAINLINE-EXIT.
231*
232*
233*
234* GET-PATIENT-INFO.
235   EXEC SQL
236     SELECT FIRSTNAME,
237       LASTNAME,
238       DATEOFBIRTH,
239       insCardNumber,
240       ADDRESS,
241       CITY,
242       POSTCODE,

```

Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.4.11 ► Click on icon or press F5 to Step Into the code until you find the statement below.

► You can move the mouse to : DB2-PATIENT-ID and see its contents

Again, notice that we get DB2 data without going to the database, but using the stub recorded earlier (Play Back file).

```
-----+-----+-----+-----+-----+-----+-----+-----+
256   FROM PATIENT
257   WHERE PATIENTID = :DB2-PATIENT-ID
258   END-EXEC.
259   Evaluate SQLCODE
260   When 0
261     MOVE '00' TO CA-RET
262   When 100
263     MOVE '01' TO CA-RET
264   When -913
265     MOVE '01' TO CA-RET
266   When Other
267     MOVE '90' TO CA-RET
+000000001
```

4.4.12 ► Keep clicking on icon or press F5 to Step Into the code until you see the bug that you had introduced.

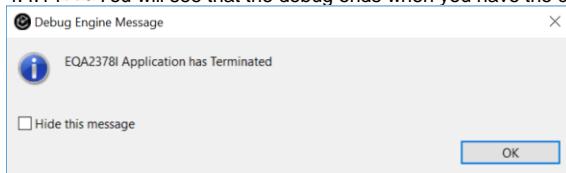
► You also can see the BUG that you introduced. Move the mouse to CA-FIRST-NAME to see the field content (Ralph), which will be replaced by "BAD NAME".

```
-----+-----+-----+-----+-----+-----+-----+-----+
270   END-Evaluate.
271   * %bug2 -- the line below will introduce a BUG
272   *
273   IF DB2-PATIENT-ID = 1
274     MOVE "BAD NAME" to CA-FIRST-NAME
275   END-IF
276   MOVE "02" to CA-NEWFI
277   *
278   EXIT.
279   *
280   *COPY HCERRSPD.
281   *=====
```

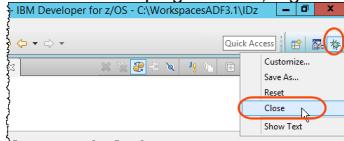
4.4.13 ► Click on icon or press F8 to resume.. Or if you prefer keep clicking on Step Into.



4.4.14 ► You will see that the debug ends when you have the dialog below. Click OK



4.4.15 ► On top right corner, right click on icon and select Close to close the debug perspective



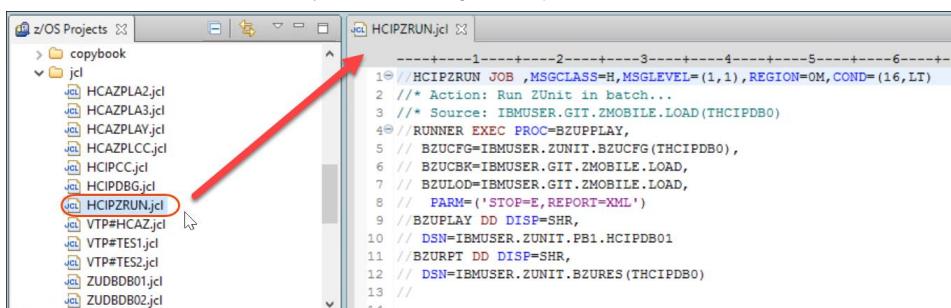
Section 5. (Optional)Run the unit test from a batch JCL.

Once a test case has been generated and built, it can be executed using a batch run via JCL.
This would enable it to be ran as part of an automated process or pipeline..

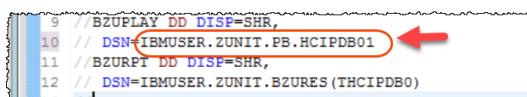
5.1 Running the unit test from a batch JCL

5.1.1 ► Close any active windows by pressing Ctrl+Shift+F4.

5.1.2 ► On the IDz z/OS Projects view, double click HCIPZRUN.jcl to open it.
This JCL will run the test case that you created using a batch job.



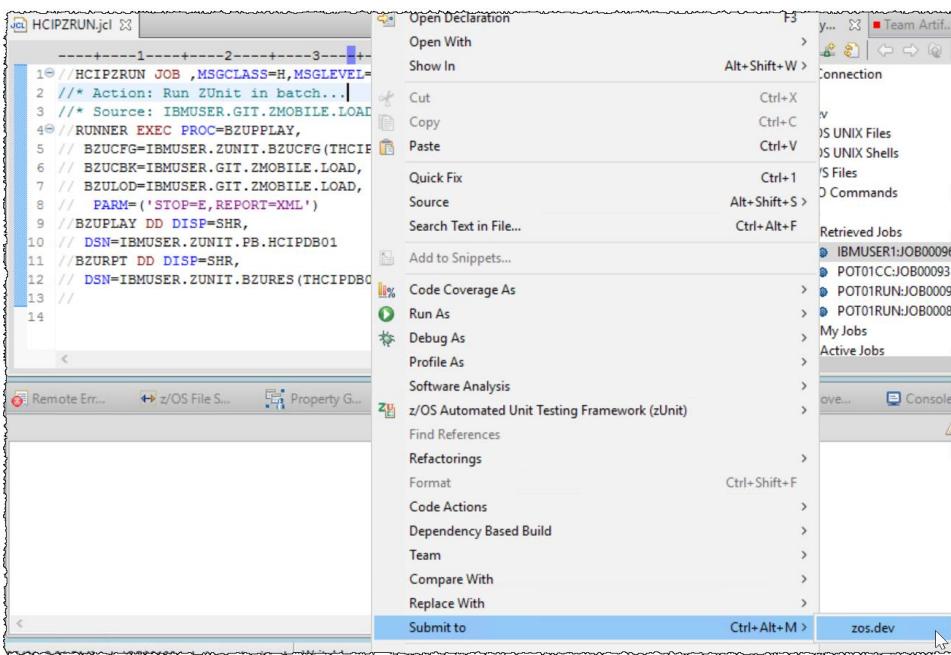
5.1.3 ► Be sure that on the line 10 the dataset name is IBMUSER.ZUNIT.PB.HCIPDB01 (instead of PB1). You may need to change this and then click Ctrl+S



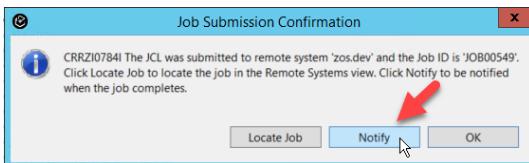
Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.1.4 ► Right click the editor and select Submit > zos.dev.

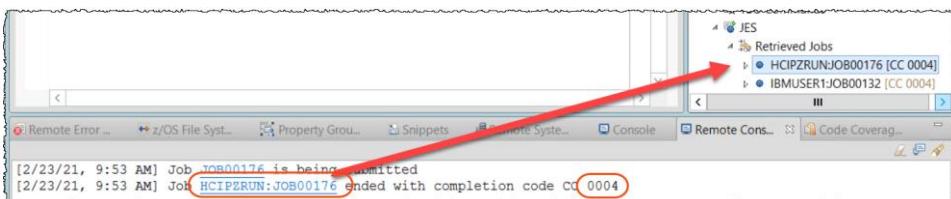


5.1.5 ► Click Notify on the Job Submission Confirmation dialog.



5.1.6 You MUST have 0004 as return code.

► Click on HCIPZRUN:JOB00xxx



Exercises Guide

Materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.1.7 ► Expand **HCIPZRUN:JOB00xxx** and verify the results double clicking on **RUNNER:REPLAY:SYSOUT**.



```
HClPZRUNJd IBMUSER.HClPZRUNJ0B00103.D0000104?.spool :1 TEST_INQ01 STARTED...2 CALL_HClPDB013 DB2_INPT ...4 DB2_OUTP ...5 CICS_OE08_HClPDB01 CHECK VALUES...6 EXEC CICS RETURN X'0000' L=002307 AREA ALLOCATED FOR RECORD COUNT:00000048 CICS_OE08_HClPDB01 SUCCESSFUL.9 ****10 AZU2001W THE TEST "INQ01" FAILED DUE TO AN ASSERTION.11 AZU1101T COMPARE FAILED IN PROCEDURE DIVISION.12 DATA ITEM NAME : CA-FIRST-NAME OF CA-PATIENT-REQUEST OF DFHCOMMAREA13 VALUE : BAD NAME14 EXPECTED VALUE: Ralph1516 TEST_INQ01 SUCCESSFUL.17
```

A red arrow points from the right margin of the code editor to the 'RUNNER:REPLAY:SYSOUT' entry in the 'Retrieved Jobs' section of the Remote Systems pane.

5.1.8 This JCL could be executed using DBB as part of a Jenkins Pipeline.

You may see an example of a groovy script used by DBB at the USS file below:

```
/var/jenkins/workspace/HealtCareAndUCD/HealthCareApp/build/ RunZUnitJCL.groovy
```

Under **zos.dev** and **z/OS UNIX Files** look for filter **groovy_samples**

```
1 import com.ibm.db2.build.CopyToHFS
2 import com.ibm.db2.build.DBBConstants
3 import com.ibm.db2.build.JCLExec
4 ****
5 * Changed Dec 27, 2019 by Regi
6 * The following sample shows how to use JCLExec API to execute a ZUnit and
7 * display the results in the console.
8 * This sample assumes that user has setup the ZUnit and a JCL to execute the
9 * ZUnit.
10 * This sample requires:
11 *   1. The data set contains the JCL.
12 *   2. The data set contains the output of the ZUnit result.
13 * Sample output:
14 *   Running ZUnit in JCL 'IBMUUSER.ZUNIT.JCL(ZUNIDB01)'
15 *   The JCL Job completed with Max-RC CC 0004
16 *   **** Module [J05CMORT] ****
17 * zUnit Test Runner 2.0.0.1 started at 2019-11-06T13:57:22.805...
18 * Test count: 1
19 * Tests passed: 1
20 * Tests failed: 0
21 * Tests in error: 0
22 *
23 ****
24 */
25/* DB2 CONF must be set for running JCLExec */
26/* def confDir = System.getenv("DB2_CONF") */
27/* added by regi - was above statement only before */
28def confdir = "/var/db2/1.0.7/conf"
29
30/* The data set contains the ZUnit JCL */
31/* For example: IBMUSER.ZUNIT.JCL */
32confDirLast = "IBMUUSER.ZUNIT.JCL"
```

zos.dev

- z/OS UNIX Files
 - My Home
 - Root
 - /var/zosconnect/servers/server30
 - groovy_samples
 - ADMIN.pw
 - BMSProcessing.groovy
 - build.groovy
 - build.properties
 - build.sh
 - CobolCompile.groovy
 - CobolCompile.groovy1.0.1
 - CobolCompile_OLD.groovy
 - Compile.groovy
 - datasets.properties
 - deploy.sh
 - deploy.groovy
 - deploy_OLD.groovy
 - file.properties
 - file.txt
 - impacts.groovy
 - JCL.groovy
 - LinkEdit.groovy
 - MFSGENUtility.groovy
 - RunZUnitJCL.groovy

Notice that this capability allows you to invoke zUnit using pipelines like Jenkins.

Congratulations! You have completed the Exercise