Z Open Development
Proof of Technology (PoT)

ZUnit-CICS Lab

Exercises Using zDT from Shares

Updated November 12, 2019

Contents

LAB 0: Unit Testing for COBOL CICS

Introduction to Unit Testing

Unit Testing is a level of software testing where a component, or individual units are tested. In the distributed world, it is quite common to find methods or functions as the individual unit of testing. By concept, unit testing also involves testing only the code – unaffected by external inputs like data base calls etc. Typically, unit testing involves stubbing out or mocking such calls so that the developer can test just the logic behind his or her code. Test stubs simulate the behavior of software components – These could be calls to a database, other functions and so on.

This lab aims at bringing Unit Testing to z/OS Applications – providing the ability to test just a single program within a CICS transaction without the need to run the whole transaction. The technology stubs out CICS calls, enabling the programs to be tested without a CICS environment and without the need to deploy to CICS after a code change. This enables a developer to test early without impacting other developers when working on a shared environment. This can then be followed by other levels of testing within the environment – like Integration testing, Regression testing and so on.

Scenario Overview

You will use z/OS Automated Unit Testing or ZUNIT which is part of IBM® Developer for Z Systems to create unit test cases for a program This is also a separate product called IBM® Z Open Unit Test which can be used with IBM® Z Open Development.

In this exercise, you will use IBM Developer for Z Systems (referred to as IDz in the remaining of this document) to create and execute automated unit tests for a program using CICS. This process includes recording or capturing data from a live CICS Application run and importing data for playback/stubbing so that you can run your unit tests without deploying.

This scenario guides you through the discovery process in roughly 30-45 minutes. By the end of the session, you'll know how to:

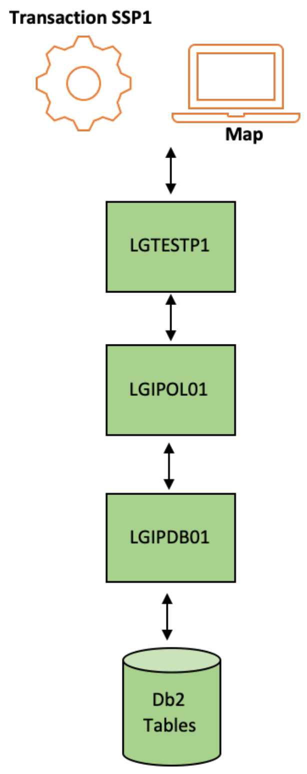
- Use ZUNIT from IBM Developer for Z systems to create a unit testing test case from a recording.
- Run the test
- Modify a program and view the failure of a test case

No previous knowledge of application architecture is needed, but experience with Eclipse-based tools and z/OS application programming are required.

The Story

GenApp is a General Insurance application based on CICS, COBOL and Db2 that can be used by Business Operations to enquire and update policy data for customers for various kinds of policies – Home, Auto etc. The scenario for this lab is based around making changes to one of the programs that is part of the transaction invoked to query a Motor Insurance Policy. And the change scenario is to ensure that a No-claim bonus of 50$ is provided to vehicles that do not have an Accident History.

The CICS transaction that is involved in this scenario is SSP1 which is the Motor Policy Enquiry transaction. This transaction consists of the programs shown below –



As a developer, you will normally go through a set of analysis steps to arrive at what constitutes a transaction – You could use simple CEDF CICS Debug facility or a Debugger or Analysis tools to help you do this. The lab assumes that you already know the program that needs a change and the aim is to create a Unit test case for this program.

Accessing the Environment

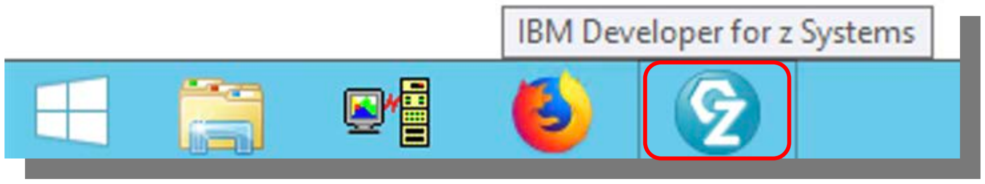
Goals

1. *Login to the Lab Environment*
2. *Access IBM Developer for Z Systems*
3. *Connect to the z/OS Environment*

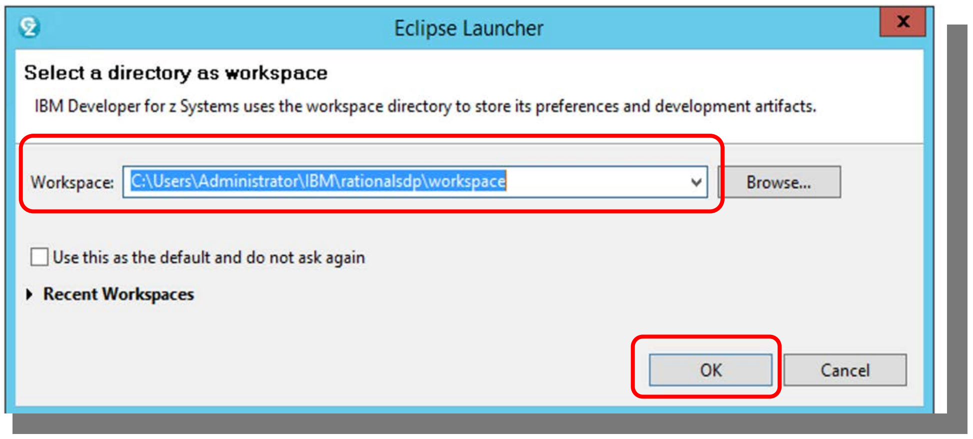
To login to the Lab Environment - Your instructor will give you the details of a remote machine which will be your lab environment. You can access it via a browser or a Remote Desktop connection. This will help you to login to the lab environment.

Steps to open IBM Developer for Z Systems

1. **Click** the **icon** for IDz highlighted **below**.



2. **Click OK** on the next screen which prompts to enter a workspace. A workspace should appear as shown below.


Eclipse Launcher

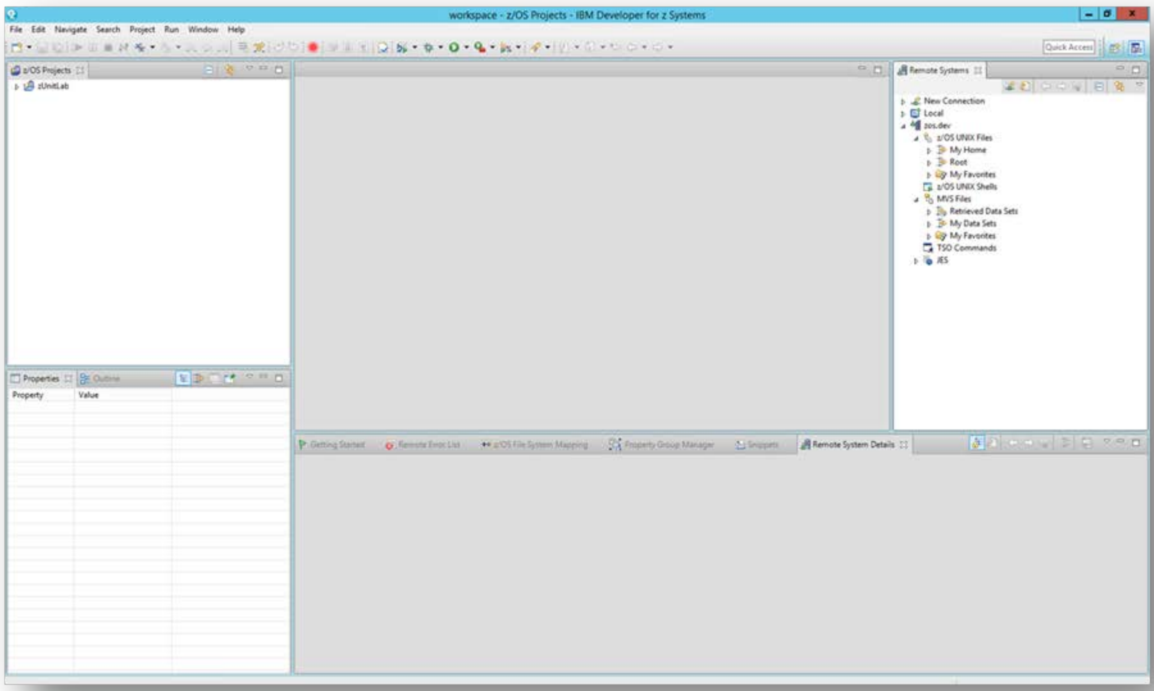
Select a directory as workspace

IBM Developer for z Systems uses the workspace directory to store its preferences and development artifacts.

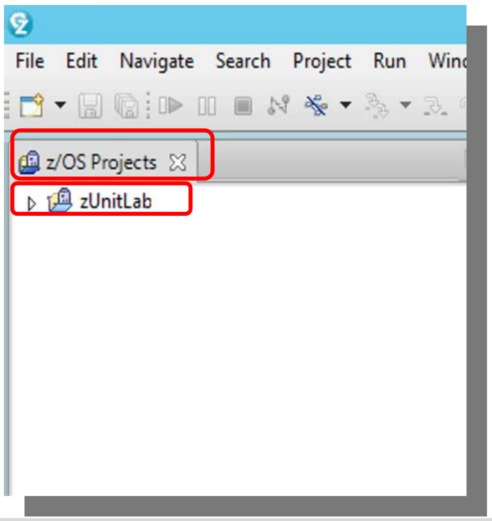
Workspace: C:\Users\Administrator\IBM\rationalsdp\workspace

If this happens to be blank or a different one, copy the following path into the workspace text box- **C:\Users\Administrator\IBM\rationalsdp\workspace** and **Click on OK**

3. This should open up IDz as shown below.



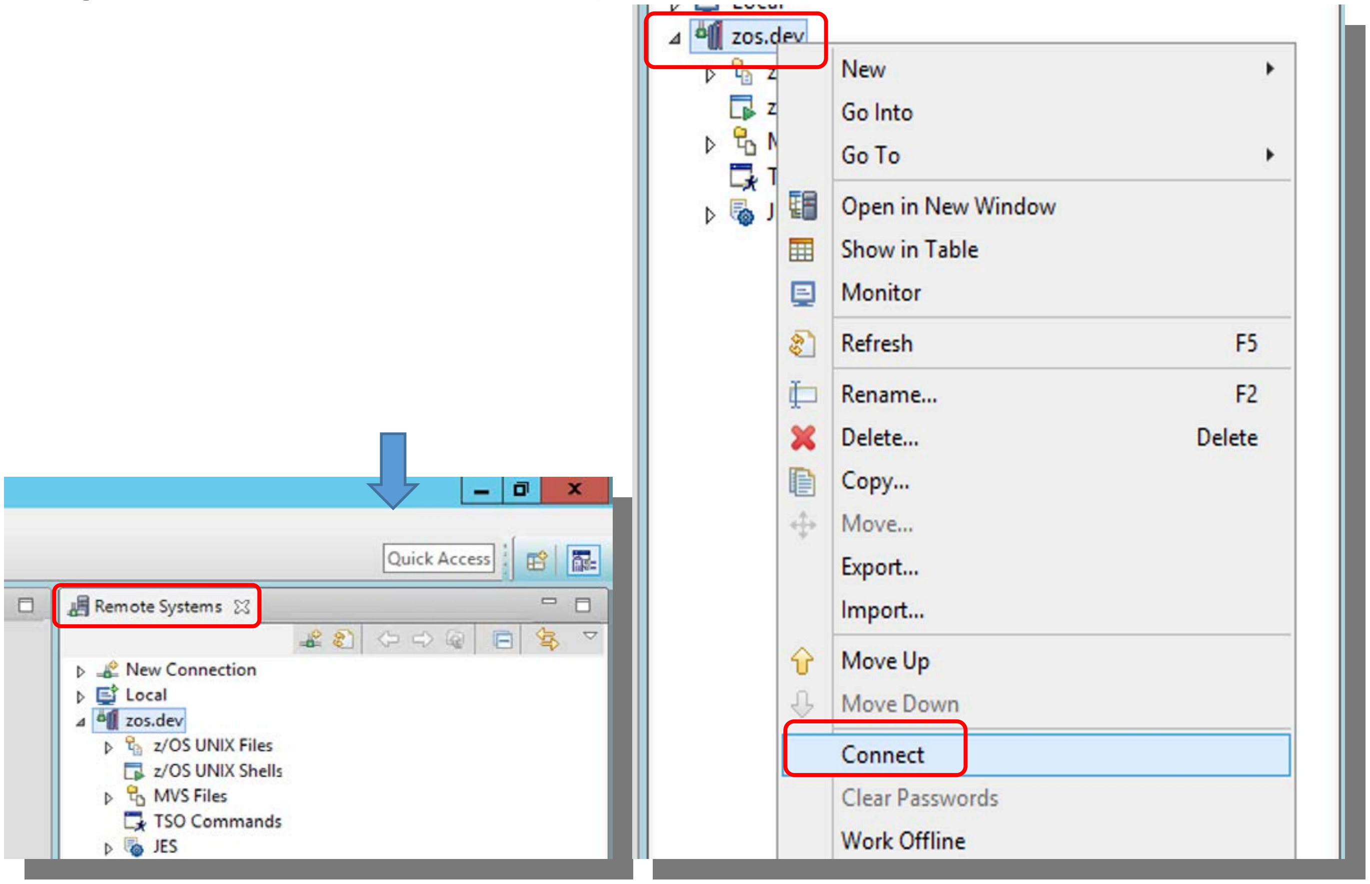
4. Note the project zUnitLab under the z/OS Projects tab that has already been setup for you.



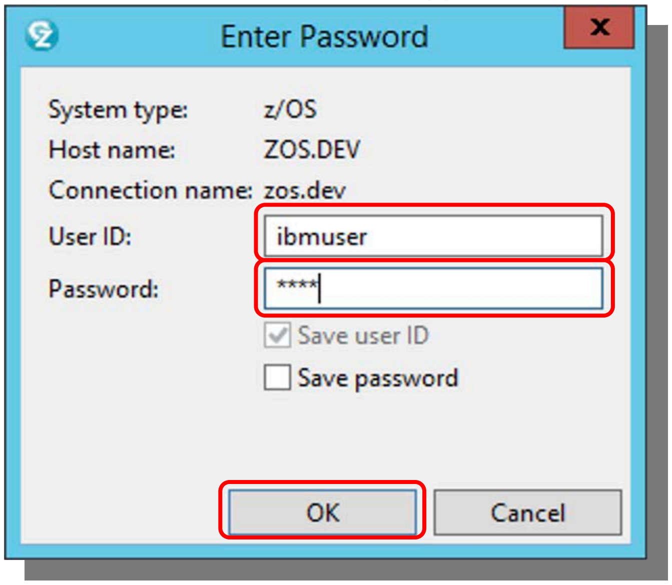
IDz helps you categorize your mainframe code (MVS and Unix) into projects and the z/OS Projects view helps you work with these projects..

Connecting to the z/OS environment

5. Right click on **zos.dev** from the **Remote systems view** and select **Connect**



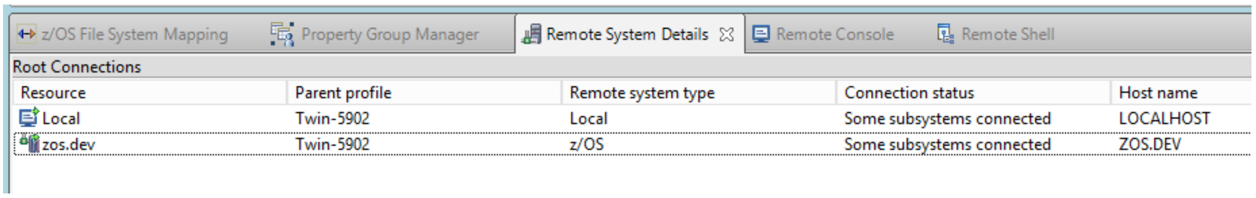
6. In the credentials window, **Type in** the user as **ibmuser** enter the password as **sys1**. **Click on OK.**


Enter Password

System type: z/OS
Host name: ZOS.DEV
Connection name: zos.dev
User ID: ibmuser
Password: ****
[✓] Save user ID
[ ] Save password

7. You are now connected to z/OS. You can view datasets, jobs. In the Remote systems view at the bottom, you can see the jobs submitted under your userid **IBMUSER .**

8. Double click and Open the **zos.dev** system.

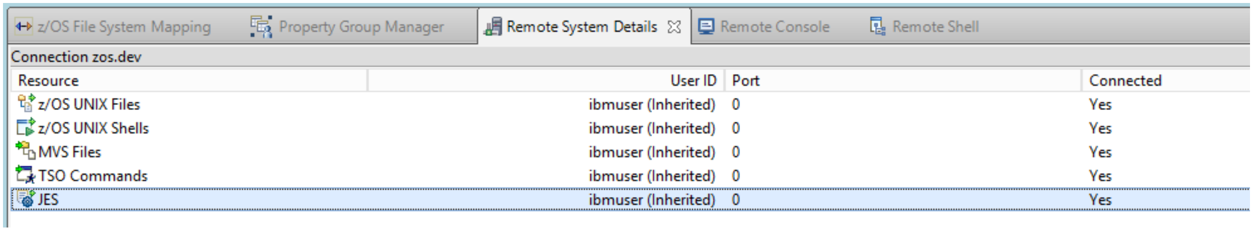


z/OS File System Mapping | Property Group Manager | Remote System Details | Remote Console | Remote Shell

Root Connections

Resource	Parent profile	Remote system type	Connection status	Host name
Local	Twin-5902	Local	Some subsystems connected	LOCALHOST
zos.dev	Twin-5902	z/OS	Some subsystems connected	ZOS.DEV

9. Double click and Open the **JES** resource

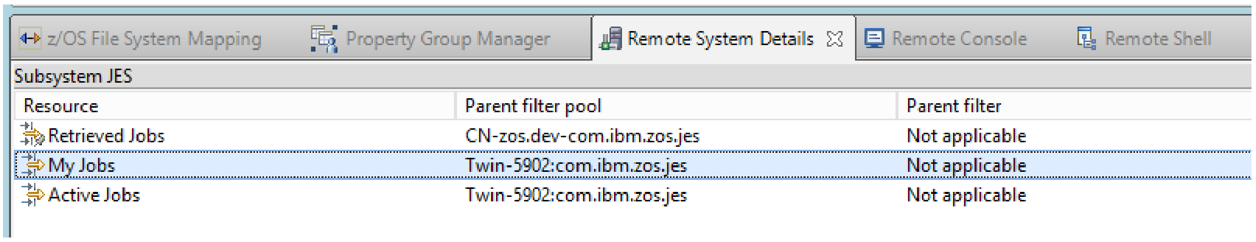


z/OS File System Mapping | Property Group Manager | Remote System Details | Remote Console | Remote Shell

Connection zos.dev

Resource	User ID	Port	Connected
z/OS UNIX Files	ibmuser (Inherited)	0	Yes
z/OS UNIX Shells	ibmuser (Inherited)	0	Yes
MVS Files	ibmuser (Inherited)	0	Yes
TSO Commands	ibmuser (Inherited)	0	Yes
JES	ibmuser (Inherited)	0	Yes

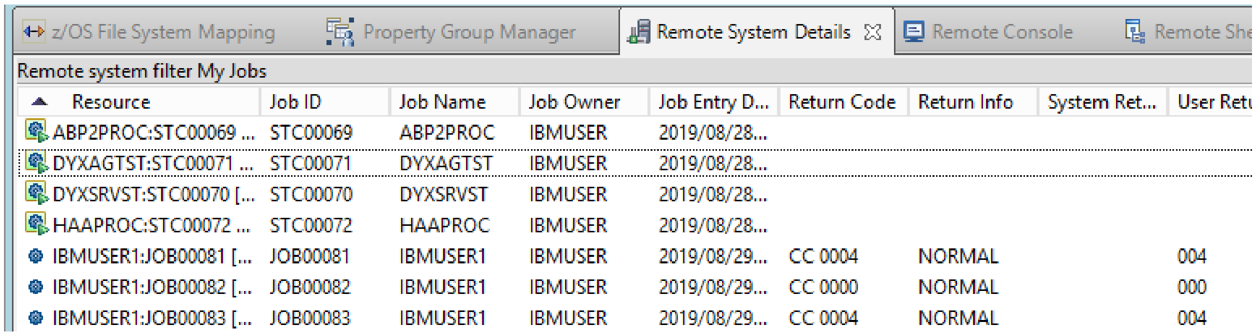
10. And Double click on **My Jobs** to show all your submitted Jobs.



z/OS File System Mapping | Property Group Manager | Remote System Details | Remote Console | Remote Shell

Subsystem JES

Resource	Parent filter pool	Parent filter
Retrieved Jobs	CN-zos.dev-com.ibm.zos.jes	Not applicable
My Jobs	Twin-5902:com.ibm.zos.jes	Not applicable
Active Jobs	Twin-5902:com.ibm.zos.jes	Not applicable



z/OS File System Mapping | Property Group Manager | Remote System Details | Remote Console | Remote She

Remote system filter My Jobs

Resource	Job ID	Job Name	Job Owner	Job Entry D...	Return Code	Return Info	System Ret...	User Retu
ABP2PROC:STC00069 ...	STC00069	ABP2PROC	IBMUSER	2019/08/28...				
DYXAGTST:STC00071 ...	STC00071	DYXAGTST	IBMUSER	2019/08/28...				
DYXSRVST:STC00070 [...	STC00070	DYXSRVST	IBMUSER	2019/08/28...				
HAAPROC:STC00072 ...	STC00072	HAAPROC	IBMUSER	2019/08/28...				
IBMUSER1:JOB00081 [...	JOB00081	IBMUSER1	IBMUSER	2019/08/29...	CC 0004	NORMAL		004
IBMUSER1:JOB00082 [...	JOB00082	IBMUSER1	IBMUSER	2019/08/29...	CC 0000	NORMAL		000
IBMUSER1:JOB00083 [...	JOB00083	IBMUSER1	IBMUSER	2019/08/29...	CC 0004	NORMAL		004

You have now successfully opened IDz and connected to the z/OS Systems.

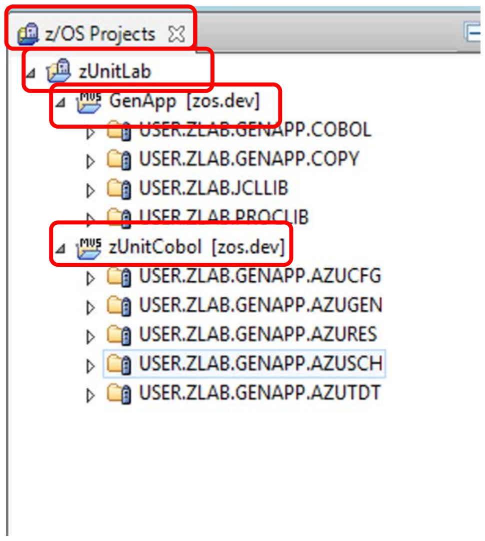
Creating a Testcase

Goals

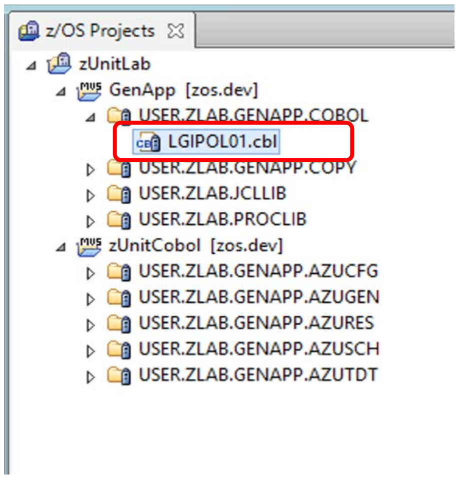
1. *View the program LGIPOL01 and understand the changes that need to be tested*
2. *Invoke the test case editor for LGIPOL01. Understand the Testcase editor*
3. *Record a test and import the test data*
4. *Create a ZUNIT test case program, run the tests and view the results of a successful test run*

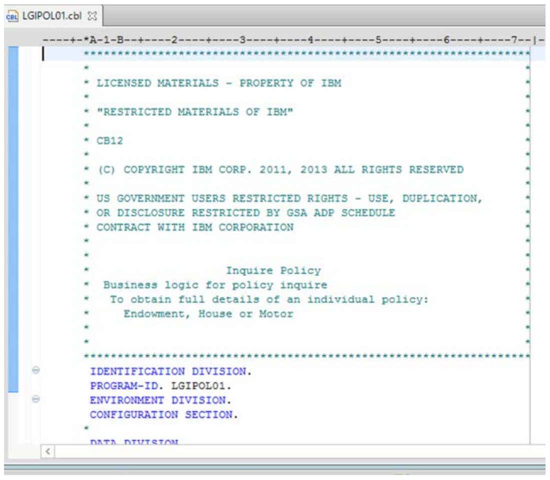
Steps

1. Expand the Project zUnitLab from the z/OS Projects view on the left by clicking on ◢



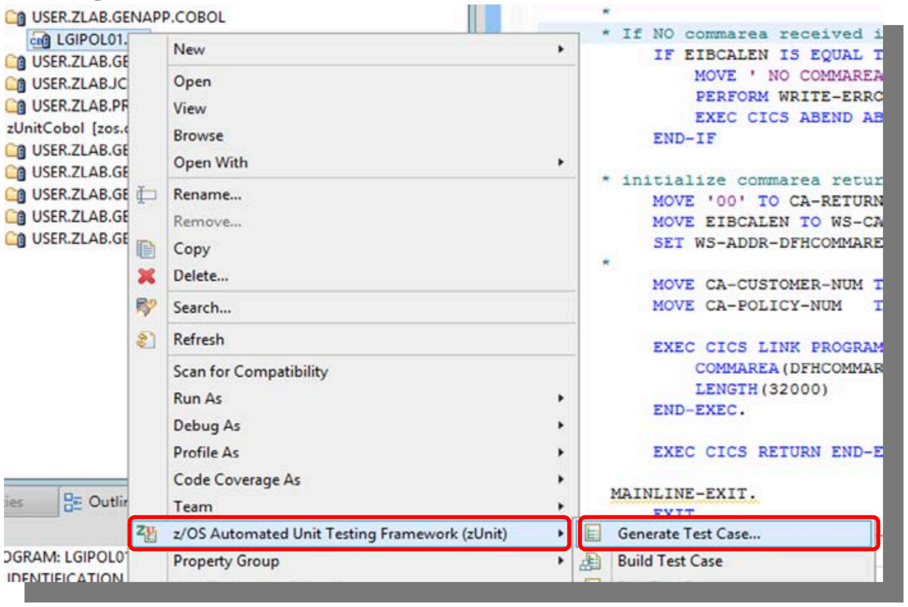
2. Double click and Open the program LGIPOL01 from **z/OS Projects > zUnitLab > GenApp > USER.ZLAB.GENAPP.COBOL > LGIPOL01.cbl** .





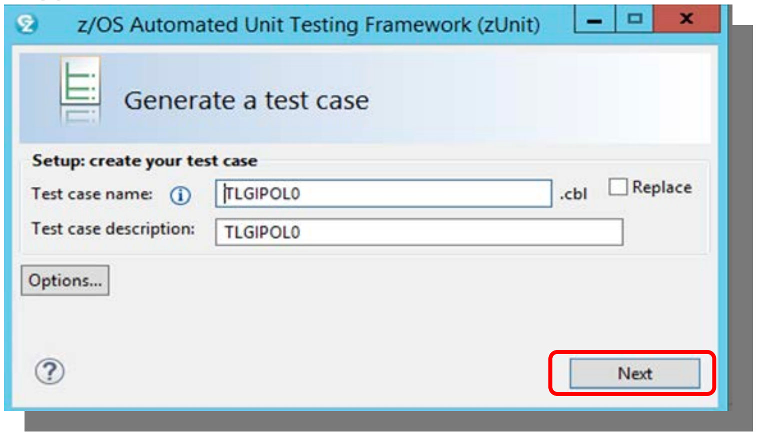
```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-
      ******************************************************************
      *                                                                *
      * LICENSED MATERIALS - PROPERTY OF IBM                           *
      *                                                                *
      * "RESTRICTED MATERIALS OF IBM"                                  *
      *                                                                *
      * CB12                                                           *
      *                                                                *
      * (C) COPYRIGHT IBM CORP. 2011, 2013 ALL RIGHTS RESERVED         *
      *                                                                *
      * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, DUPLICATION,      *
      * OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE                   *
      * CONTRACT WITH IBM CORPORATION                                  *
      *                                                                *
      *                                                                *
      *                    Inquire Policy                              *
      *  Business logic for policy inquire                             *
      *   To obtain full details of an individual policy:              *
      *     Endowment, House or Motor                                  *
      *                                                                *
      *                                                                *
      ******************************************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID. LGIPOL01.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
      *
       DATA DIVISION
```

3. To create a testcase for LGIPOL01, Right click on the program and select **z/OS Automated Unit Testing Framework(zUnit) > Generate Test Case..**



```
      * If NO commarea received i
        IF EIBCALEN IS EQUAL T
            MOVE ' NO COMMAREA
            PERFORM WRITE-ERRO
            EXEC CICS ABEND AB
        END-IF

      * initialize commarea retur
        MOVE '00' TO CA-RETURN
        MOVE EIBCALEN TO WS-CA
        SET WS-ADDR-DFHCOMMARE
      *
        MOVE CA-CUSTOMER-NUM T
        MOVE CA-POLICY-NUM   T

        EXEC CICS LINK PROGRAM
            COMMAREA(DFHCOMMAR
            LENGTH(32000)
        END-EXEC.

        EXEC CICS RETURN END-E

    MAINLINE-EXIT.
```

4. This opens a window where you can name your testcase. Accept the auto-generated name ZUnit suggests - TLGIPOL01. **Click on Next.**

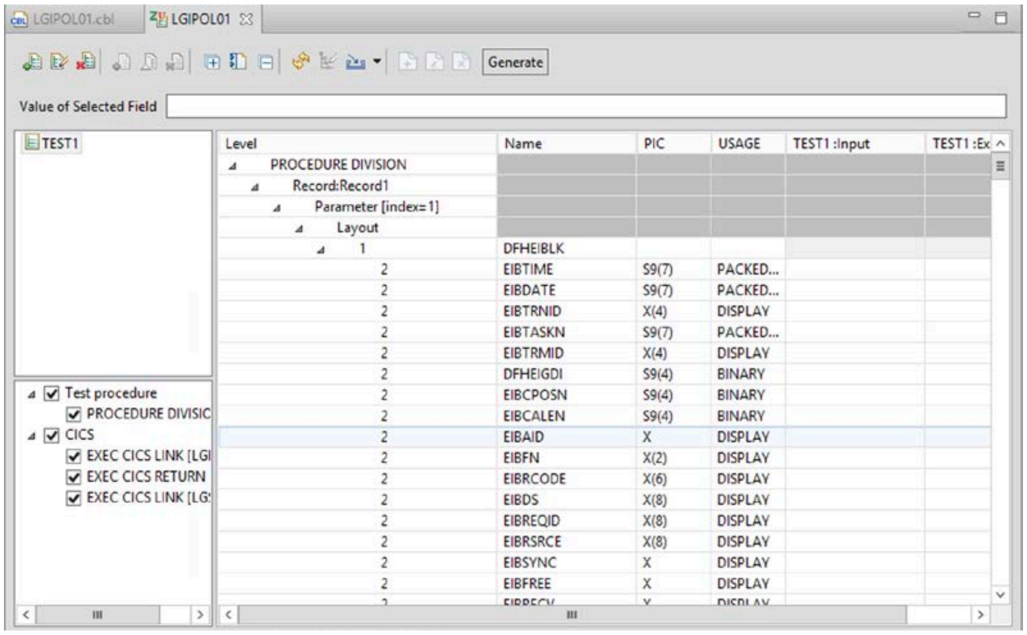


Setup: create your test case

Test case name: TLGIPOL0 .cbl ☐ Replace

Test case description: TLGIPOL0

5. The Test Case Editor is opened up on the screen.



Value of Selected Field

TEST1

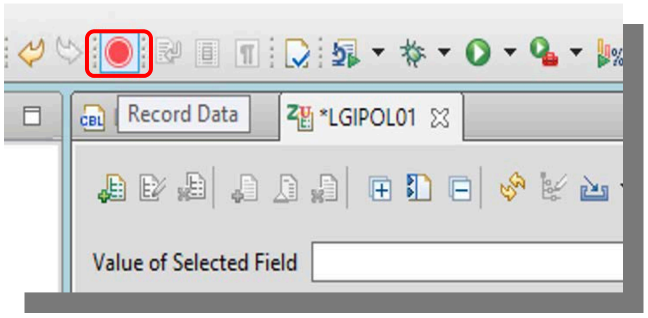
Level	Name	PIC	USAGE	TEST1 :Input	TEST1 :Ex
PROCEDURE DIVISION					
Record:Record1					
Parameter [index=1]					
Layout					
1	DFHEIBLK				
2	EIBTIME	S9(7)	PACKED...		
2	EIBDATE	S9(7)	PACKED...		
2	EIBTRNID	X(4)	DISPLAY		
2	EIBTASKN	S9(7)	PACKED...		
2	EIBTRMID	X(4)	DISPLAY		
2	DFHEIGDI	S9(4)	BINARY		
2	EIBCPOSN	S9(4)	BINARY		
2	EIBCALEN	S9(4)	BINARY		
2	EIBAID	X	DISPLAY		
2	EIBFN	X(2)	DISPLAY		
2	EIBRCODE	X(6)	DISPLAY		
2	EIBDS	X(8)	DISPLAY		
2	EIBREQID	X(8)	DISPLAY		
2	EIBRSRCE	X(8)	DISPLAY		
2	EIBSYNC	X	DISPLAY		
2	EIBFREE	X	DISPLAY		

- ✓ Test procedure
  - ✓ PROCEDURE DIVISIC
- ✓ CICS
  - ✓ EXEC CICS LINK [LGI
  - ✓ EXEC CICS RETURN
  - ✓ EXEC CICS LINK [LG:

Understanding the test case editor

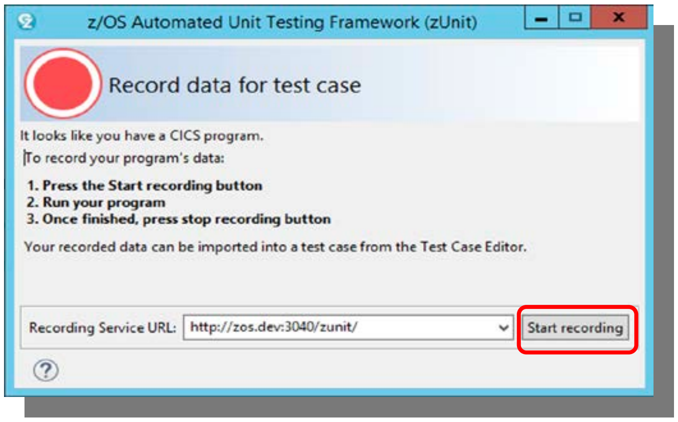
- The bottom left box summarizes all the input output variable structures – In this example, the program has a COMMAREA exchanged via the PROCEDURE DIVISION and three EXEC CICS statements.
- The COMMAREA and the CICS statements are also accompanied by the CICS DFHEIBLK variables.
- The variables that are an input to the program – e.g. certain commarea values on entry, DFHEIBLK on entry are to be entered against the TEST1:Input column.
- The variables that are returned by the program are the output from the program logic that a developer should be checking. The expected results of these variables should be in the TEST1:Expected column

6. To import data from a live-run into the test case, click on the Record button in IDz



For a developer working on a complex CICS program it is difficult to understand what values go into each of the Input variables and which go into the output. It will be helpful if the developer has to verify only the variables that matter to his/her code change. This is made easier with the Record function. This allows you to capture the data that is passed through the program during an actual live run.

In the dialog box that opens up, click on **Start Recording**. The recording service url points to the CICS region from where the live run is captured. For this lab, the url does not need any changes.


z/OS Automated Unit Testing Framework (zUnit)
Record data for test case

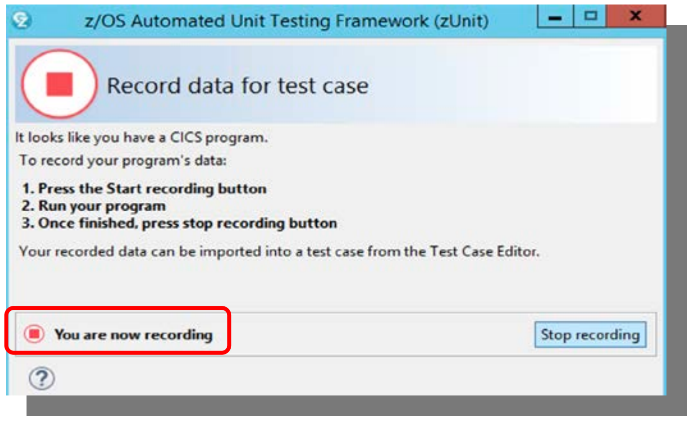
It looks like you have a CICS program.
To record your program's data:

1. Press the Start recording button
2. Run your program
3. Once finished, press stop recording button

Your recorded data can be imported into a test case from the Test Case Editor.

Recording Service URL: http://zos.dev:3040/zunit/

When recording is turned on, the message ‘**You are now recording**’ appears.


z/OS Automated Unit Testing Framework (zUnit)
Record data for test case

It looks like you have a CICS program.
To record your program's data:

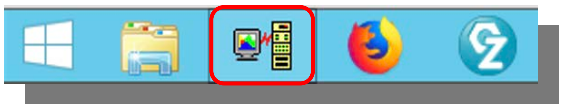
1. Press the Start recording button
2. Run your program
3. Once finished, press stop recording button

Your recorded data can be imported into a test case from the Test Case Editor.

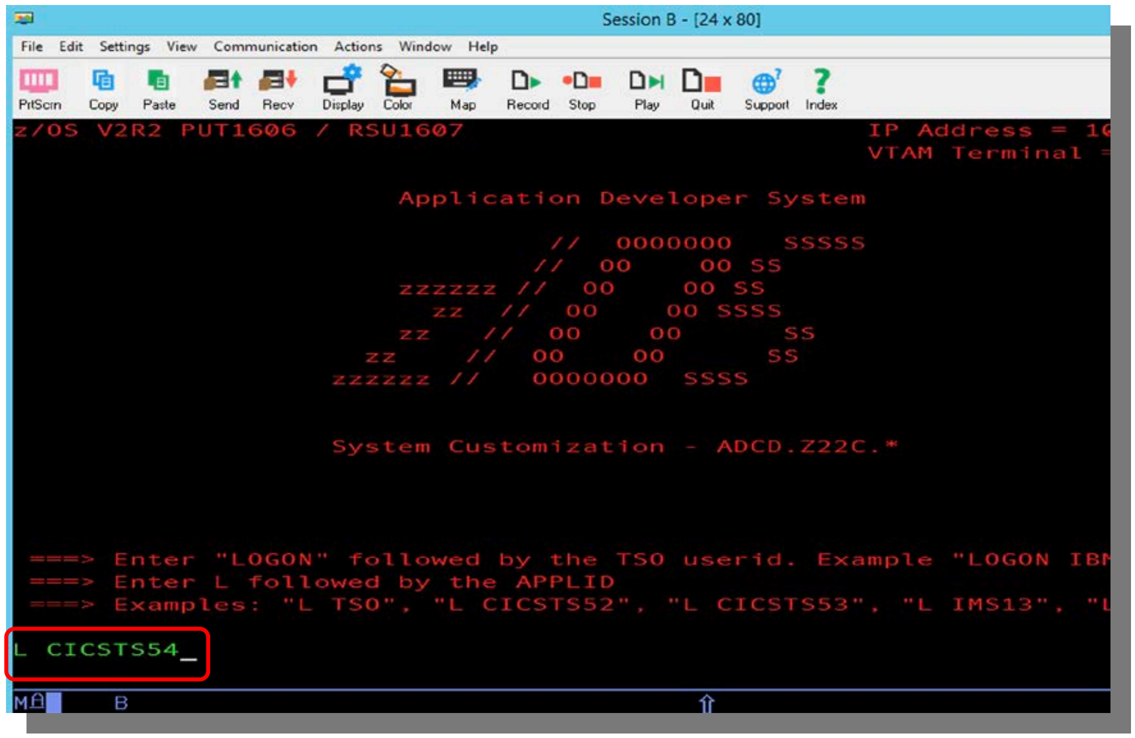
You are now recording

You are now free to invoke your CICS transaction in the manner that it is usually invoked – via a green screen or a webpage etc. In this lab, we will sign on to a host emulator to trigger the transaction SSP1.

7. Click on the host emulator icon at the bottom of the screen

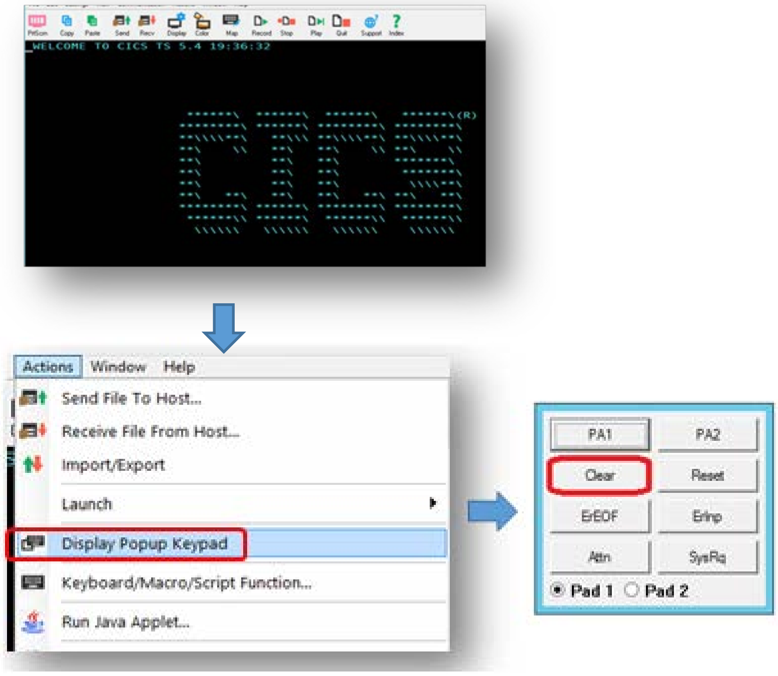


8. This opens the host emulator. Type **L CICSTS54** in the bottom of the screen.


Session B - [24 x 80]
File Edit Settings View Communication Actions Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit Support Index
MA B

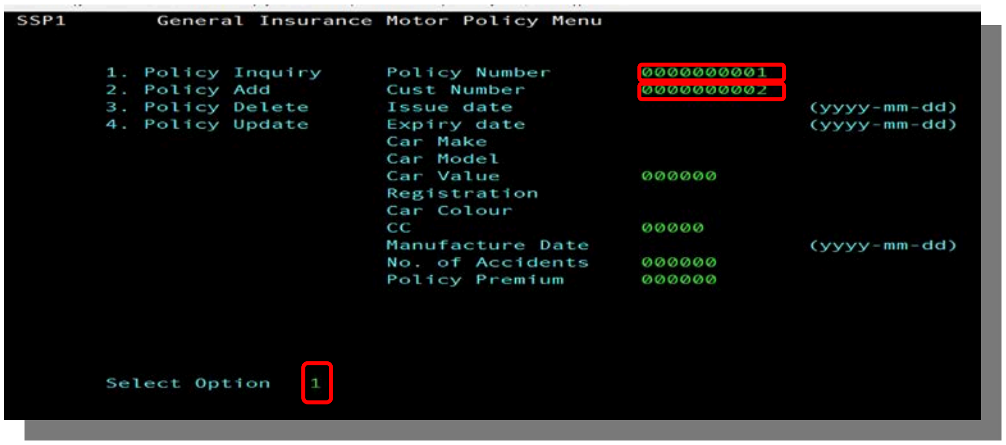
```
z/OS V2R2 PUT1606 / RSU1607                             IP Address = 10
                                                        VTAM Terminal =

                       Application Developer System

                                //  OOOOOOO   SSSSS
                               //  OO    OO SS
                       zzzzzz //  OO    OO SS
                         zz  //  OO    OO SSSS
                       zz   //  OO    OO      SS
                     zz    //  OO    OO      SS
                   zzzzzz //   OOOOOOO  SSSS


                   System Customization - ADCD.Z22C.*



 ===> Enter "LOGON" followed by the TSO userid. Example "LOGON IBM
 ===> Enter L followed by the APPLID
 ===> Examples: "L TSO", "L CICSTS52", "L CICSTS53", "L IMS13", "L

L CICSTS54_
```

9. Clear the welcome screen by right-clicking and choosing **Display Keypad** and then selecting **Clear**.



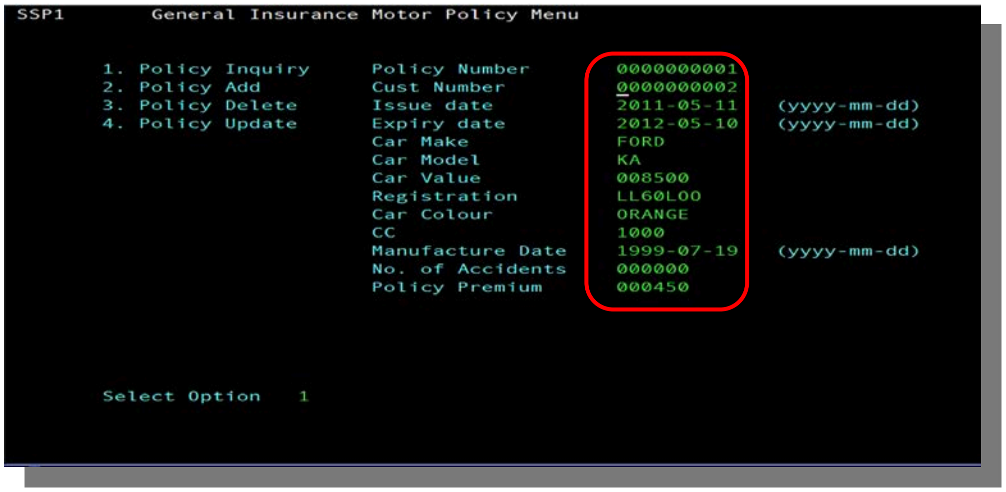
```
WELCOME TO CICS TS 5.4 19:36:32
```

10. Type SSP1. This brings up the Motor Policy screen. Type **0000000001** against the **Policy Number**, **0000000002** against the **Customer Number**, Enter '**1**' against the **Option** and Press **Enter/Ctrl. Use Tab to navigate to the different fields.**



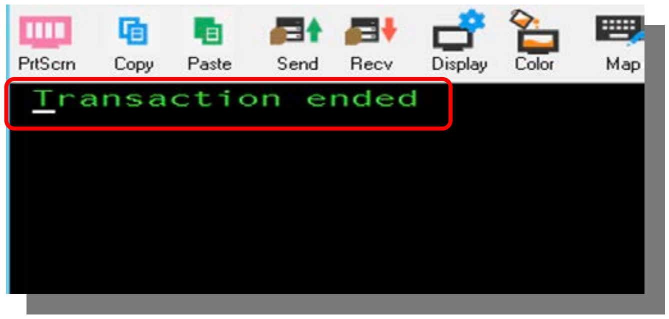
```
SSP1          General Insurance Motor Policy Menu

        1. Policy Inquiry       Policy Number        0000000001
        2. Policy Add           Cust Number          0000000002
        3. Policy Delete        Issue date                          (yyyy-mm-dd)
        4. Policy Update        Expiry date                         (yyyy-mm-dd)
                                Car Make
                                Car Model
                                Car Value            000000
                                Registration
                                Car Colour
                                CC                   00000
                                Manufacture Date                    (yyyy-mm-dd)
                                No. of Accidents     000000
                                Policy Premium       000000




        Select Option   1
```

11. This brings up the Motor policy screen for the customer number 000000001. Please note that the No. of Accidents for this customer is 0.



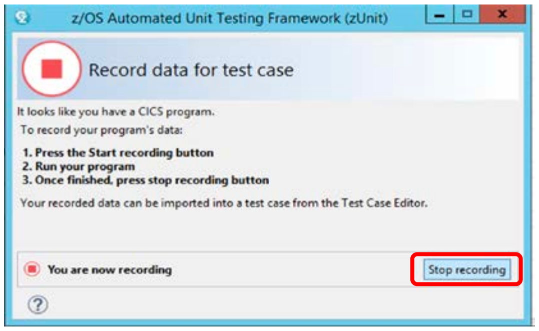
```
SSP1          General Insurance Motor Policy Menu

        1. Policy Inquiry       Policy Number        0000000001
        2. Policy Add           Cust Number          0000000002
        3. Policy Delete        Issue date           2011-05-11     (yyyy-mm-dd)
        4. Policy Update        Expiry date          2012-05-10     (yyyy-mm-dd)
                                Car Make             FORD
                                Car Model            KA
                                Car Value            008500
                                Registration         LL60LOO
                                Car Colour           ORANGE
                                CC                   1000
                                Manufacture Date     1999-07-19     (yyyy-mm-dd)
                                No. of Accidents     000000
                                Policy Premium       000450




        Select Option   1
```

12. Press **F3** to exit from the transaction. The **Transaction ended** message is displayed.



```
Transaction ended
```

13. Switch back to IDz. Click Stop on the recording window that is open. The message 'Test Data is imported successfully' is displayed. Click **OK** on the Import Test Data window.


z/OS Automated Unit Testing Framework (zUnit)

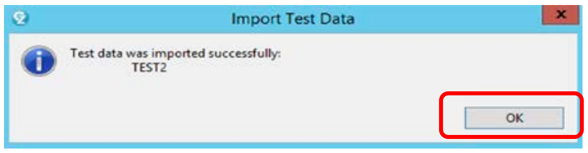
Record data for test case

It looks like you have a CICS program.
To record your program's data:

1. Press the Start recording button
2. Run your program
3. Once finished, press stop recording button

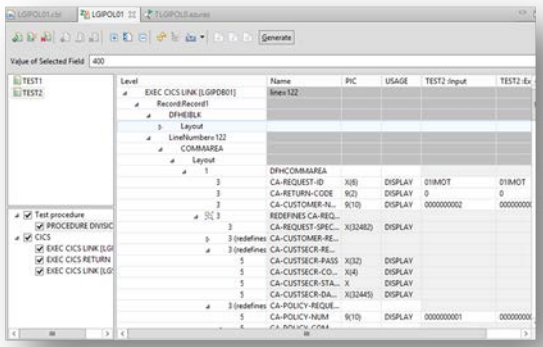
Your recorded data can be imported into a test case from the Test Case Editor.

You are now recording


Import Test Data

Test data was imported successfully:
TEST2

14. You can now see a new test – TEST2 created in the editor and it is populated with values from the live run. Use the scroll bar to view the data that is populated. You can see that this is the same that was displayed in the CICS screen.

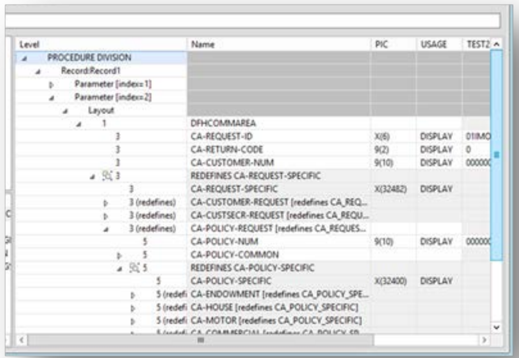


Value of Selected Field 400

TEST1
TEST2

Test procedure
PROCEDURE DIVISIC
CICS
EXEC CICS LINK [LGI
EXEC CICS RETURN
EXEC CICS LINK [LG

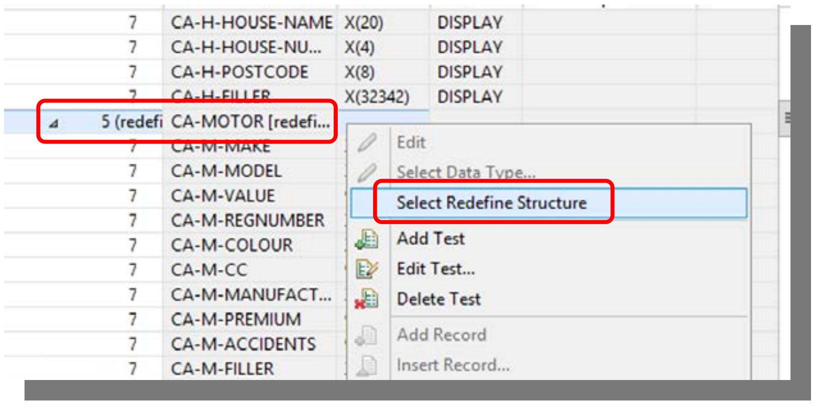
Level	Name	PIC	USAGE	TEST2 :input	TEST2 :Ex
EXEC CICS LINK [LGIPDB01]	line=122				
Record:Record1					
DFHEIBLK					
Layout					
LineNumber=122					
COMMAREA					
Layout					
1	DFHCOMMAREA				
3	CA-REQUEST-ID	X(6)	DISPLAY	01IMOT	01IMOT
3	CA-RETURN-CODE	9(2)	DISPLAY	0	0
3	CA-CUSTOMER-N...	9(10)	DISPLAY	0000000002	00000000
3	REDEFINES CA-REQ...				
3	CA-REQUEST-SPEC...	X(32482)	DISPLAY		
3 (redefines	CA-CUSTOMER-RE...				
3 (redefines	CA-CUSTSECR-RE...				
5	CA-CUSTSECR-PASS	X(32)	DISPLAY		
5	CA-CUSTSECR-CO...	X(4)	DISPLAY		
5	CA-CUSTSECR-STA...	X	DISPLAY		
5	CA-CUSTSECR-DA...	X(32445)	DISPLAY		
3 (redefines	CA-POLICY-REQUE...				
5	CA-POLICY-NUM	9(10)	DISPLAY	0000000001	00000000


Level	Name	PIC	USAGE	TEST2
PROCEDURE DIVISION				
Record:Record1				
Parameter [index=1]				
Parameter [index=2]				
Layout				
1	DFHCOMMAREA			
3	CA-REQUEST-ID	X(6)	DISPLAY	01IMO
3	CA-RETURN-CODE	9(2)	DISPLAY	0
3	CA-CUSTOMER-NUM	9(10)	DISPLAY	00000
3	REDEFINES CA-REQUEST-SPECIFIC			
3	CA-REQUEST-SPECIFIC	X(32482)	DISPLAY	
3 (redefines)	CA-CUSTOMER-REQUEST [redefines CA_REQ...			
3 (redefines)	CA-CUSTSECR-REQUEST [redefines CA_REQU...			
3 (redefines)	CA-POLICY-REQUEST [redefines CA_REQUES...			
5	CA-POLICY-NUM	9(10)	DISPLAY	00000
5	CA-POLICY-COMMON			
5	REDEFINES CA-POLICY-SPECIFIC			
5	CA-POLICY-SPECIFIC	X(32400)	DISPLAY	
5 (redefi	CA-ENDOWMENT [redefines CA_POLICY_SPE...			
5 (redefi	CA-HOUSE [redefines CA_POLICY_SPECIFIC]			
5 (redefi	CA-MOTOR [redefines CA_POLICY_SPECIFIC]			

The program LGIPOL01 is a program that is in the call chain of the transaction SSP1. It calls the program LGIPDB01 and passed values to its caller via the commarea. Hence the values in the commarea are the expected output. The call to program LGIPDB01 is populated as Input as it is an input to LGIPOL01.

You can also choose to delete the testcase TEST1 as it is blank.

15. ZUNIT also gives you the option to select the right layout in case of a multi layout copybook as the one used here. To record the Motor policy layout, scroll down and Right click the CA-MOTOR redefinition variable under **PROCEDURE DIVISION > Record:Record1 > Parameter[index=2] > Layout > DFHCOMMAREA > CA-MOTOR -> Select Redefine Structure**.

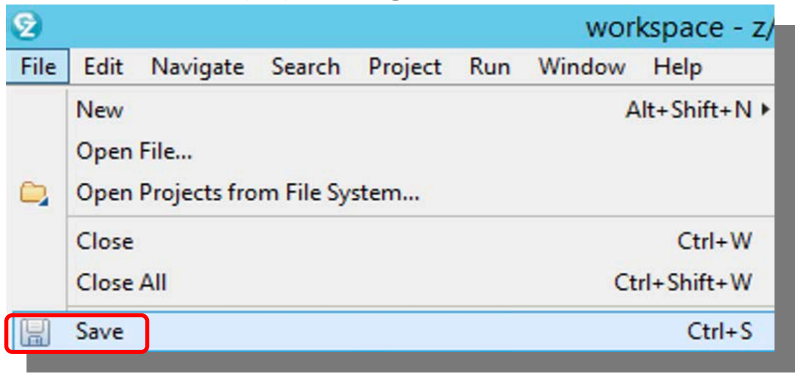


```
7          CA-H-HOUSE-NAME  X(20)     DISPLAY
7          CA-H-HOUSE-NU...  X(4)     DISPLAY
7          CA-H-POSTCODE     X(8)     DISPLAY
7          CA-H-FILLER       X(32342) DISPLAY
5 (redefi  CA-MOTOR [redefi...
7          CA-M-MAKE
7          CA-M-MODEL
7          CA-M-VALUE
7          CA-M-REGNUMBER
7          CA-M-COLOUR
7          CA-M-CC
7          CA-M-MANUFACT...
7          CA-M-PREMIUM
7          CA-M-ACCIDENTS
7          CA-M-FILLER
```

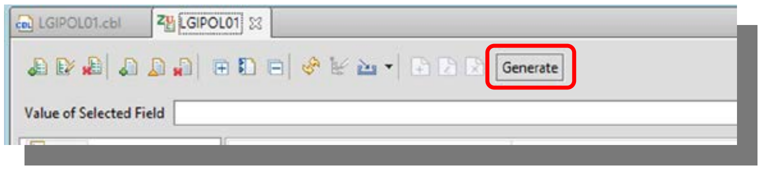
The motor policy will now be correctly mapped to the layout in the commarea.

16. Repeat step #15 for the motor policy layout returned by the LINK call to LGIPDB01. This can be found under **EXEC CICS LINK [LGIPDB01] > Record:Record1 > Line Number – 122 > COMMAREA > Layout > DFHCOMMAREA > CA-MOTOR**.

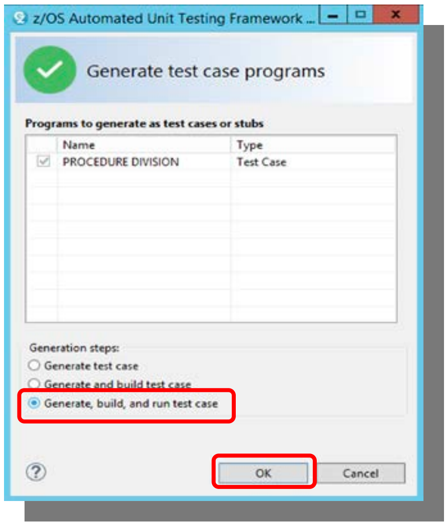
17. Save the Test Entry by clicking on Ctrl+S or File > Save



18. Now that the expected output values and the input has been set in the editor from the recorded run, **Click on Generate** to generate the test case program.



19. In the pop-up window, Select the option **Generate, build and run test case** and **Click on OK**.


z/OS Automated Unit Testing Framework ...

Generate test case programs

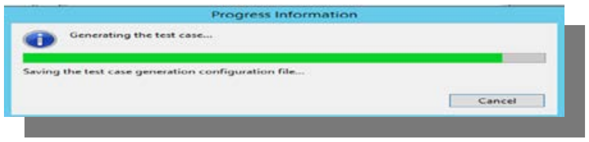
Programs to generate as test cases or stubs

	Name	Type
✓	PROCEDURE DIVISION	Test Case

Generation steps:

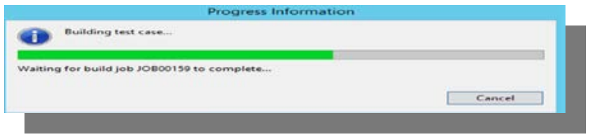
- ○ Generate test case
- ○ Generate and build test case
- ● Generate, build, and run test case

This action creates a test case program with two tests (TEST1 is blank) and a stub program that is to stub out CICS. It also builds the programs and runs the test case


Progress Information

Generating the test case...

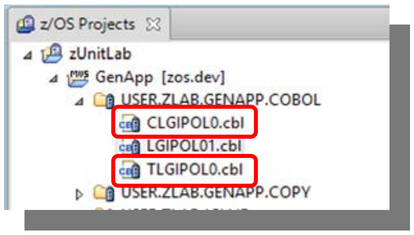
Saving the test case generation configuration file...


Progress Information

Building test case...

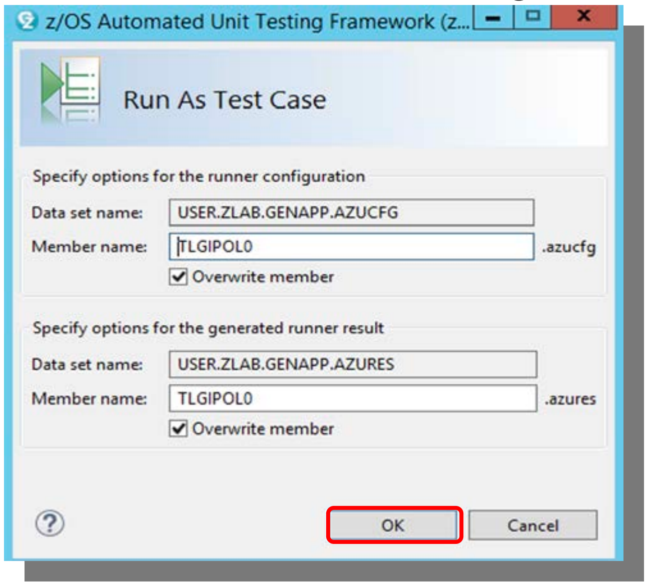
Waiting for build job JOB00159 to complete...

The two programs created can be viewed in the program dataset


z/OS Projects

- zUnitLab
  - GenApp [zos.dev]
    - USER.ZLAB.GENAPP.COBOL
      - CLGIPOL0.cbl
      - LGIPOL01.cbl
      - TLGIPOL0.cbl
    - USER.ZLAB.GENAPP.COPY

20. Click next on the default test run configuration. The default option runs all the tests in the testcase


z/OS Automated Unit Testing Framework (z...

Run As Test Case

Specify options for the runner configuration

Data set name: USER.ZLAB.GENAPP.AZUCFG

Member name: TLGIPOL0 .azucfg

☑ Overwrite member

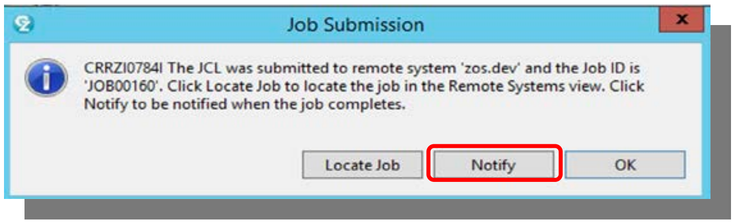
Specify options for the generated runner result

Data set name: USER.ZLAB.GENAPP.AZURES

Member name: TLGIPOL0 .azures

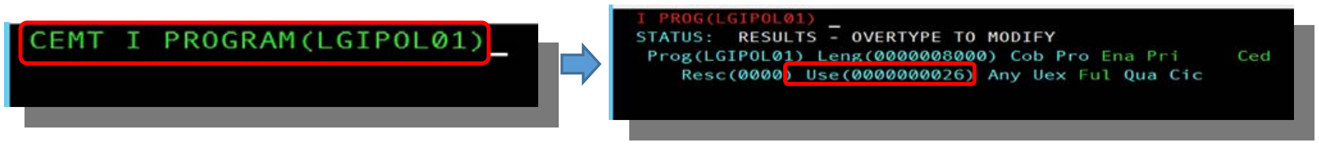
☑ Overwrite member

21. Click notify on the popup. The test case is now running as a batch job with CICS stubbed out. The changed program is not yet deployed into CICS.


Job Submission

CRRZI0784I The JCL was submitted to remote system 'zos.dev' and the Job ID is 'JOB00160'. Click Locate Job to locate the job in the Remote Systems view. Click Notify to be notified when the job completes.

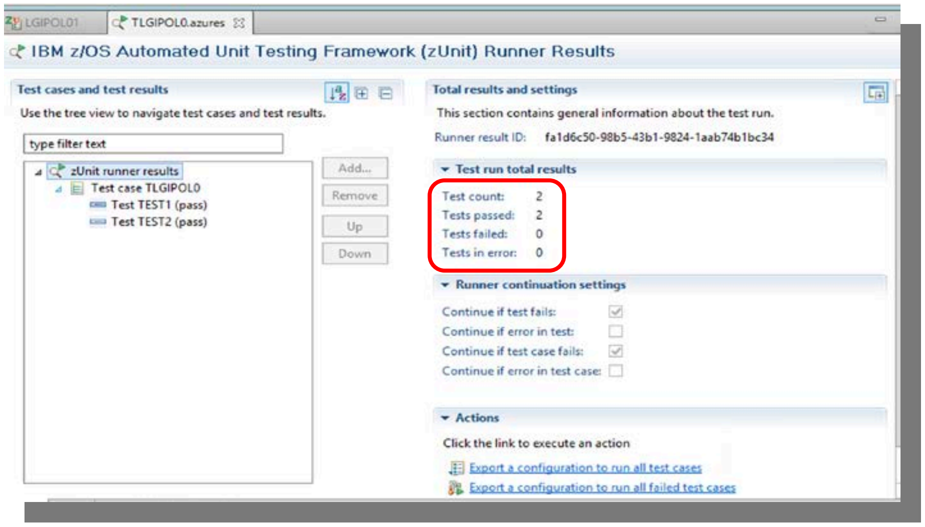
***Optional Step** – You may verify that CICS is stubbed out by running a CEMT I PROGRAM(LGIPOL01) in the CICS region before and after the test case run. The Count will remain the same as the program is not running within CICS when the test case is executed showing how CICS programs that are in the call chain of the transaction can be individually tested without the need to deploy into CICS*


```
CEMT I PROGRAM(LGIPOL01)_
```

```
I PROG(LGIPOL01)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Prog(LGIPOL01) Leng(0000008000) Cob Pro Ena Pri     Ced
    Resc(0000) Use(0000000026) Any Uex Ful Qua Cic
```

While running CEMT, please take care that the transaction is not triggered directly on CICS as it will increase the count.

22. After a successful run, the results screen is displayed where you can see that the two test cases have passed.



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

Test cases and test results

Use the tree view to navigate test cases and test results.

type filter text

- zUnit runner results
  - Test case TLGIPOL0
    - Test TEST1 (pass)
    - Test TEST2 (pass)

Total results and settings

This section contains general information about the test run.

Runner result ID: fa1d6c50-98b5-43b1-9824-1aab74b1bc34

Test run total results

Test count:	2
Tests passed:	2
Tests failed:	0
Tests in error:	0

Runner continuation settings

Continue if test fails:	✓
Continue if error in test:	
Continue if test case fails:	✓
Continue if error in test case:	

Actions

Click the link to execute an action

Export a configuration to run all test cases

Export a configuration to run all failed test cases

You have now created a test case with data imported from a recorded run and you have been able to successfully test a CICS program without the need of a CICS environment.

Modifying Source code and Viewing a failed test case

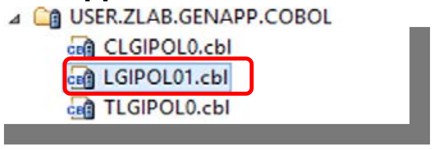
Goals

1. *Edit the business rules in the program LGIPOL01*
2. *Run the testcase and view the failed testcase*

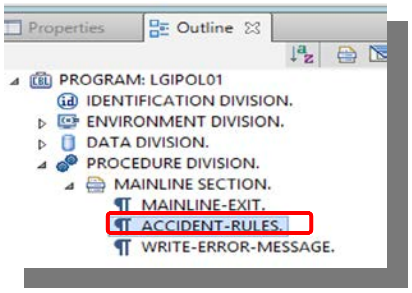
3. Correct the test data and run the test case again

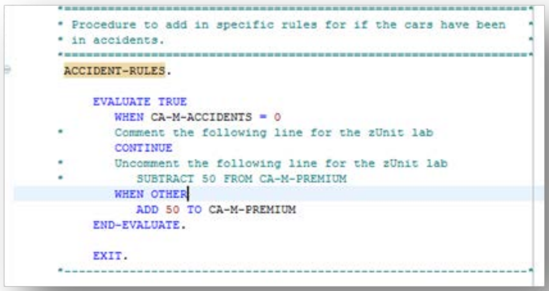
Steps

1. Switch to the program LGIPOL01. To do this, Double click on the program name from **z/OS Projects > GenApp > USER.ZLAB.GENAPP.COBOL**



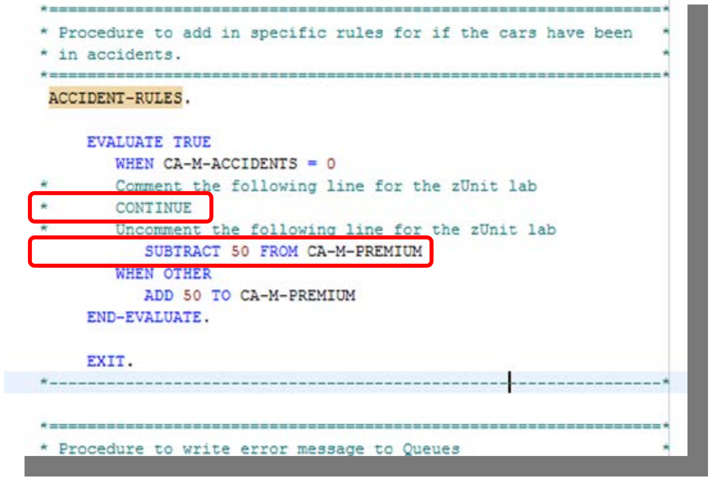
2. From the Outline view at the bottom left of IDz, Navigate to the paragraph **ACCIDENT-RULES**





```
*=================================================================*
* Procedure to add in specific rules for if the cars have been    *
* in accidents.                                                   *
*=================================================================*
 ACCIDENT-RULES.

     EVALUATE TRUE
        WHEN CA-M-ACCIDENTS = 0
*       Comment the following line for the zUnit lab
        CONTINUE
*       Uncomment the following line for the zUnit lab
*          SUBTRACT 50 FROM CA-M-PREMIUM
        WHEN OTHER
           ADD 50 TO CA-M-PREMIUM
     END-EVALUATE.

     EXIT.
*-----------------------------------------------------------------*
```

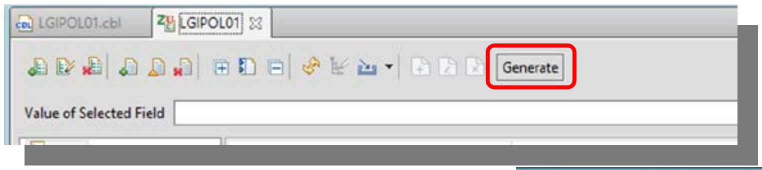
3. To make the code change for no-claim bonus, the developer has to subtract 50$ from the Policy premium if the car was accident free i.e. the number of accidents is 0. To do this, **comment the line CONTINUE** and **uncomment the line SUBTRACE 50 FROM CA-M-PREMIUM**

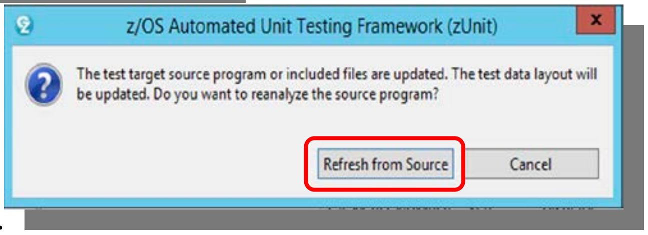


```
*=================================================================*
* Procedure to add in specific rules for if the cars have been    *
* in accidents.                                                   *
*=================================================================*
 ACCIDENT-RULES.

     EVALUATE TRUE
        WHEN CA-M-ACCIDENTS = 0
*       Comment the following line for the zUnit lab
*       CONTINUE
*       Uncomment the following line for the zUnit lab
           SUBTRACT 50 FROM CA-M-PREMIUM
        WHEN OTHER
           ADD 50 TO CA-M-PREMIUM
     END-EVALUATE.

     EXIT.
*-----------------------------------------------------------------*

*=================================================================*
* Procedure to write error message to Queues                      *
```

4. Save the program with File > Save or Ctrl+S.

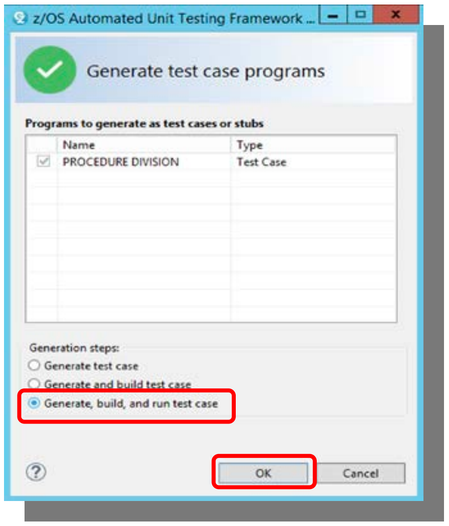
5. Click Generate on the Test case Editor.




z/OS Automated Unit Testing Framework (zUnit)

The test target source program or included files are updated. The test data layout will be updated. Do you want to reanalyze the source program?

6. Select Refresh from Source on the pop-up.
7. In the pop-up window, Select the option **Generate, build and run test case** and **Click on OK**.


z/OS Automated Unit Testing Framework

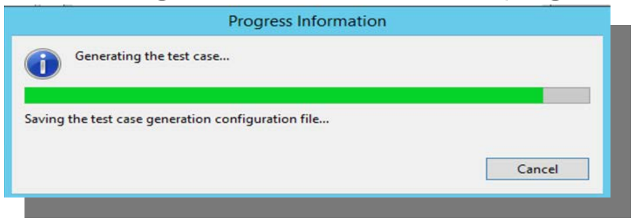
Generate test case programs

Programs to generate as test cases or stubs

Name	Type
PROCEDURE DIVISION	Test Case

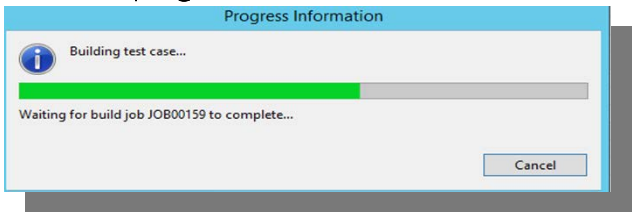
Generation steps:
- ( ) Generate test case
- ( ) Generate and build test case
- (●) Generate, build, and run test case

This action generates a new test case program, re-builds the programs and runs the test case.


Progress Information

Generating the test case...

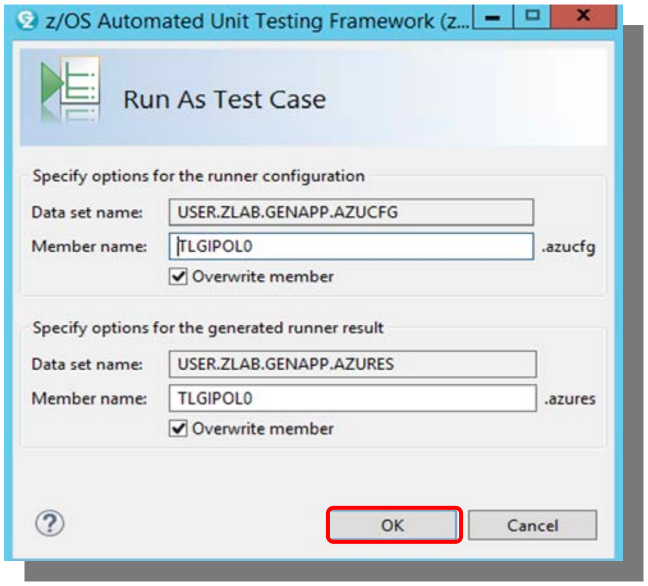
Saving the test case generation configuration file...


Progress Information

Building test case...

Waiting for build job JOB00159 to complete...

8. Click next on the default test run configuration. The default option runs all the tests in the testcase



z/OS Automated Unit Testing Framework (z...

Run As Test Case

Specify options for the runner configuration

Data set name: USER.ZLAB.GENAPP.AZUCFG

Member name: TLGIPOL0 .azucfg

☑ Overwrite member

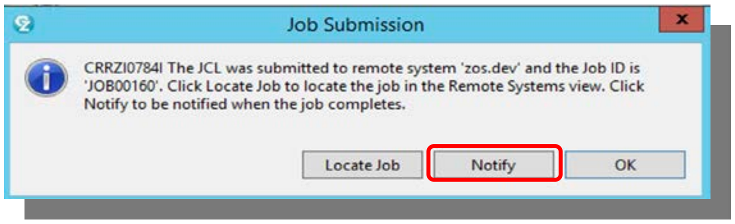
Specify options for the generated runner result

Data set name: USER.ZLAB.GENAPP.AZURES

Member name: TLGIPOL0 .azures

☑ Overwrite member

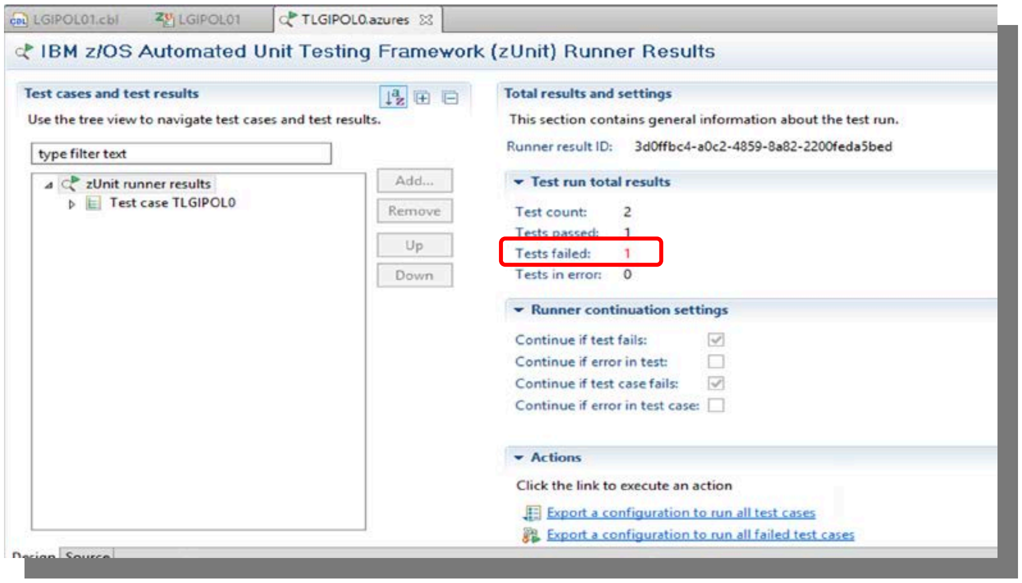
9. Click notify on the popup. The test case is now running as a batch job with CICS stubbed out. The changed program is not yet deployed into CICS.



Job Submission

CRRZI0784I The JCL was submitted to remote system 'zos.dev' and the Job ID is 'JOB00160'. Click Locate Job to locate the job in the Remote Systems view. Click Notify to be notified when the job completes.

10. This time the result will show a failure. As we did not change the expected output to match the new value with the no-claim bonus.



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

Test cases and test results

Use the tree view to navigate test cases and test results.

type filter text

zUnit runner results
Test case TLGIPOL0

Total results and settings

This section contains general information about the test run.

Runner result ID: 3d0ffbc4-a0c2-4859-8a82-2200feda5bed

Test run total results

Test count:	2
Tests passed:	1
Tests failed:	1
Tests in error:	0

Runner continuation settings

Continue if test fails:	✓
Continue if error in test:	
Continue if test case fails:	✓
Continue if error in test case:	

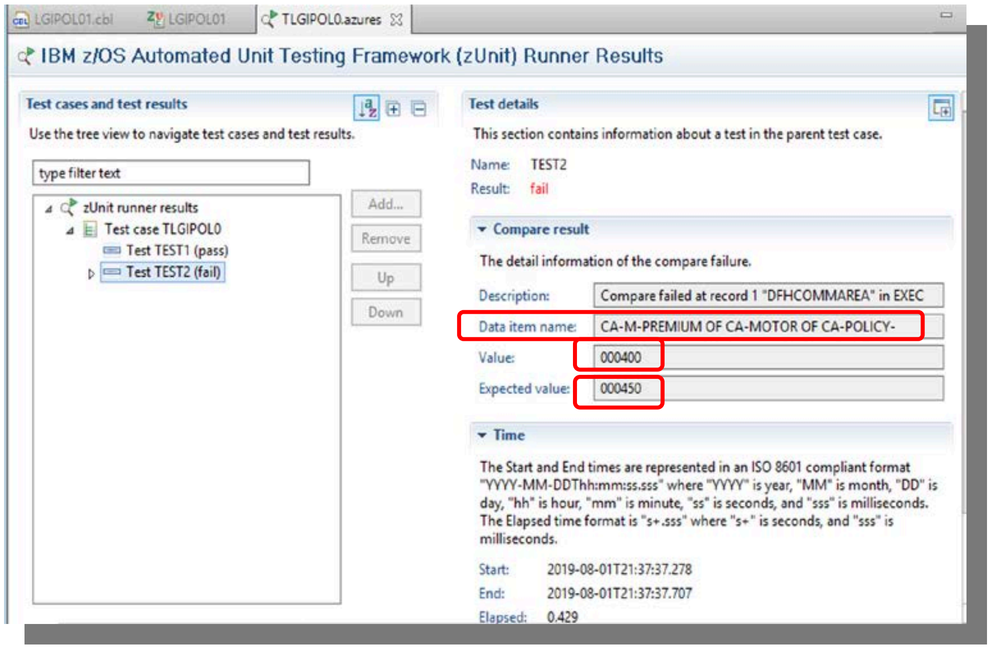
Actions

Click the link to execute an action

Export a configuration to run all test cases

Export a configuration to run all failed test cases

11. Expand on zUnit runner results > Test case TLGIPOL0 > TEST2 to view the exact failure reason.



LGIPOL01.cbl | LGIPOL01 | TLGIPOL0.azures

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

Test cases and test results

Use the tree view to navigate test cases and test results.

type filter text

- zUnit runner results
  - Test case TLGIPOL0
    - Test TEST1 (pass)
    - Test TEST2 (fail)

Test details

This section contains information about a test in the parent test case.

Name: TEST2
Result: fail

Compare result

The detail information of the compare failure.

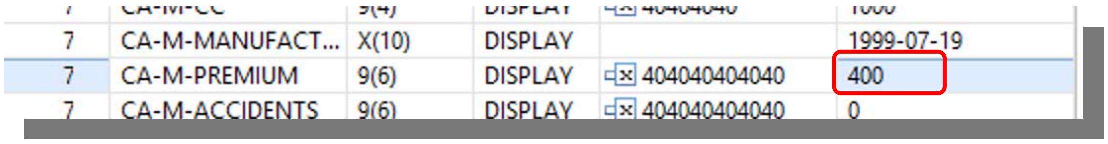
Description: Compare failed at record 1 "DFHCOMMAREA" in EXEC
Data item name: CA-M-PREMIUM OF CA-MOTOR OF CA-POLICY-
Value: 000400
Expected value: 000450

Time

The Start and End times are represented in an ISO 8601 compliant format "YYYY-MM-DDThh:mm:ss.sss" where "YYYY" is year, "MM" is month, "DD" is day, "hh" is hour, "mm" is minute, "ss" is seconds, and "sss" is milliseconds. The Elapsed time format is "s+.sss" where "s+" is seconds, and "sss" is milliseconds.

Start: 2019-08-01T21:37:37.278
End: 2019-08-01T21:37:37.707
Elapsed: 0.429

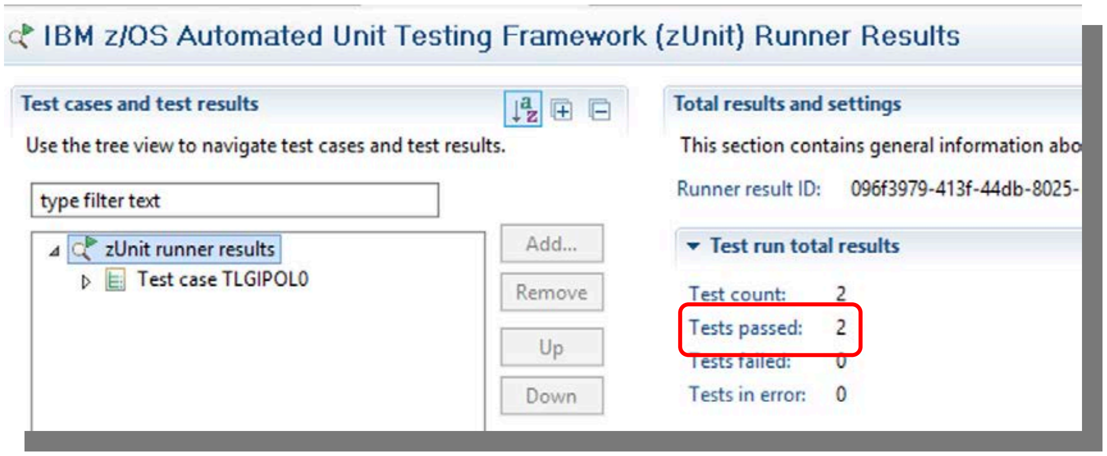
12. To correct this, switch to the Test entry editor. Change the Expected value column in **PROCEDURE DIVISION > Record:Record1 > Parameter[index=2] > Layout > DFHCOMMAREA > CA-MOTOR > CA-M-PREMIUM** to **400** as this includes the no-claim bonus deduction.



7	CA-M-MANUFACT...	X(10)	DISPLAY		1999-07-19
7	CA-M-PREMIUM	9(6)	DISPLAY	404040404040	400
7	CA-M-ACCIDENTS	9(6)	DISPLAY	404040404040	0

13. Repeat the steps #5 and #6 above to re-run the test case.

14. This time, the results show that the test cases have passed.



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

Test cases and test results

Use the tree view to navigate test cases and test results.

type filter text

- zUnit runner results
  - Test case TLGIPOL0

Total results and settings

This section contains general information abo

Runner result ID: 096f3979-413f-44db-8025-

Test run total results

Test count: 2
Tests passed: 2
Tests failed: 0
Tests in error: 0

You have now learnt how a failed test case is viewed in ZUNIT.

Running zUnit from the command line

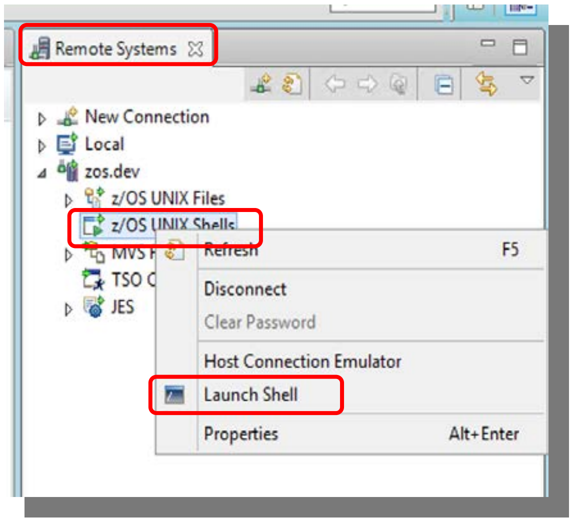
Goals

1. *Invoke zUnit from the command line with the new run configuration*

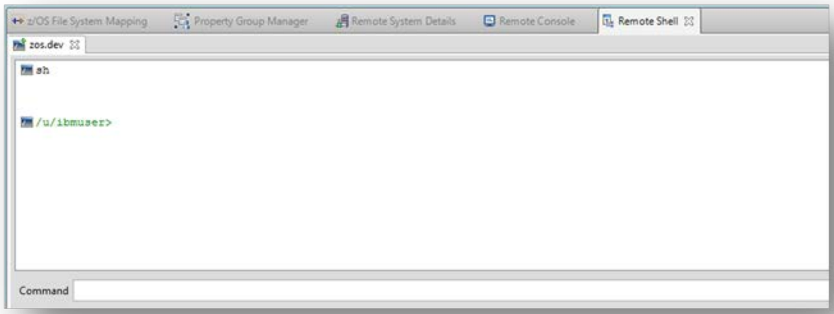
One of the mandates of a delivery pipeline is to be able to run automated unit tests from a pipeline. This section shows how to invoke zUnit from a USS(z/OS Unix) command line. This is useful when unit tests have to run as part of a pipeline run by tools like Jenkins.

Steps

1. From the **Remote Systems View**, **Right click on z/OS Unix Shells** > Select **Launch Shell**

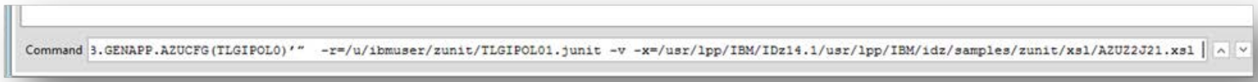


2. This displays the Remote shell at the bottom of IDz**.**



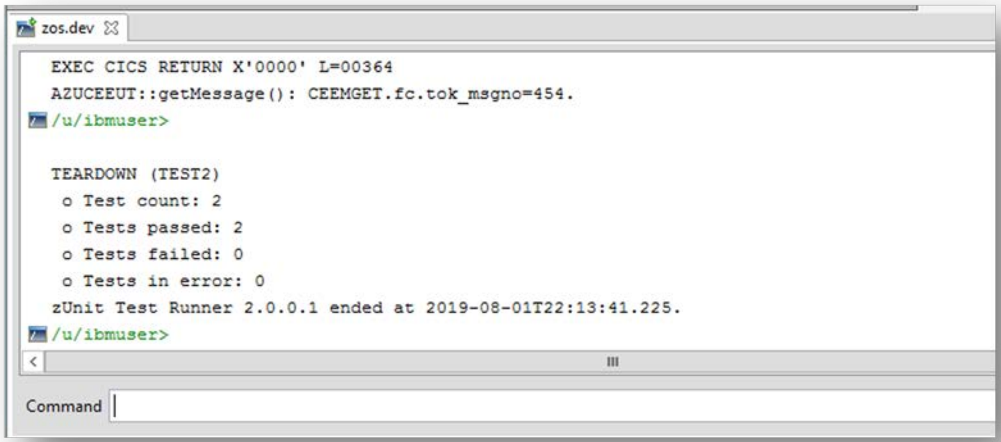
```
sh

/u/ibmuser>
```

3. Type the following in the **Command** text box. Hit **Enter**.
zunit -c="//'USER.ZLAB.GENAPP.AZUCFG(TLGIPOL0)'" -r=/u/ibmuser/zunit/TLGIPOL01.junit -v -x=/usr/lpp/IBM/IDz14.1/usr/lpp/IBM/idz/samples/zunit/xsl/AZUZ2J21.xsl



```
3.GENAPP.AZUCFG(TLGIPOL0)'"  -r=/u/ibmuser/zunit/TLGIPOL01.junit -v -x=/usr/lpp/IBM/IDz14.1/usr/lpp/IBM/idz/samples/zunit/xsl/AZUZ2J21.xsl
```

4. The test case is executed, and the results are displayed in the Remote shell



zos.dev

```
  EXEC CICS RETURN X'0000' L=00364
  AZUCEEUT::getMessage(): CEEMGET.fc.tok_msgno=454.
/u/ibmuser>

  TEARDOWN (TEST2)
   o Test count: 2
   o Tests passed: 2
   o Tests failed: 0
   o Tests in error: 0
  zUnit Test Runner 2.0.0.1 ended at 2019-08-01T22:13:41.225.
/u/ibmuser>
```

You have now executed a zUnit test case from a command line.

Conclusion

As part of this lab, you have created a Unit Test case for a CICS COBOL program that is a part of a transaction. You have also run the test case and tested this program by stubbing out CICS – without the need to deploy into a CICS environment enabling early testing without affecting other developers in a shared environment.