

IBM Watsonx Code Assistant for Z: Hands On Lab

1. Purpose of the document

The purpose of this document is to showcase the capability of IBM watsonx Code Assistant for Z to execute Understand, Refactor and Transform phases on Z Virtual Access (zVA) with a Cobol-Java Batch project using **Live watson AI Code Transformation Model**.

This document contains set of instructions, which if followed step by step will give enough knowledge and confidence to use IBM watsonx Code Assistant for Z in any project. Changes in instructions sequence execution may be required when used in project depending on the environment setup.

IBM watsonx Code Assistant for Z Refactoring Assistant can be used by:

- Architects
- Business Analysts
- Developers

This document will help anyone to use the IBM watsonx Code Assistant for Z, which helps developers identify the part of the application to refactor into modular and reusable services.

2. Setup

1. Get your access for workshop link provided prior to the workshop.

IBM Z Trial

Welcome to IBM Z Trial.

Thank you for choosing to try out ADDI-Watsonx Code Assists for Z . You can find your user credentials in the last email we sent you.

This demo will be active for 3 days from the date of the same email.

If you have any problems or you wish to extend the trial please contact ztrial@uk.ibm.com

Username

Username

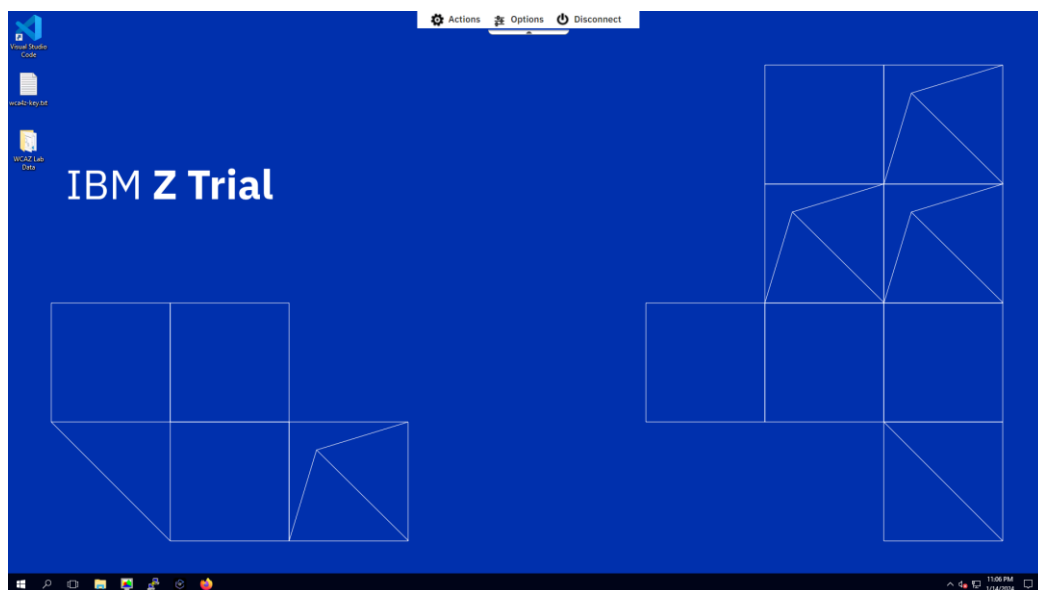
Password

Password

Cancel

Sign in

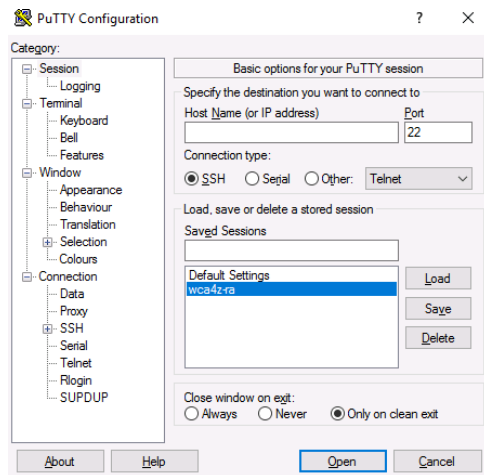
2. Login with your credentials to get into ZVA.



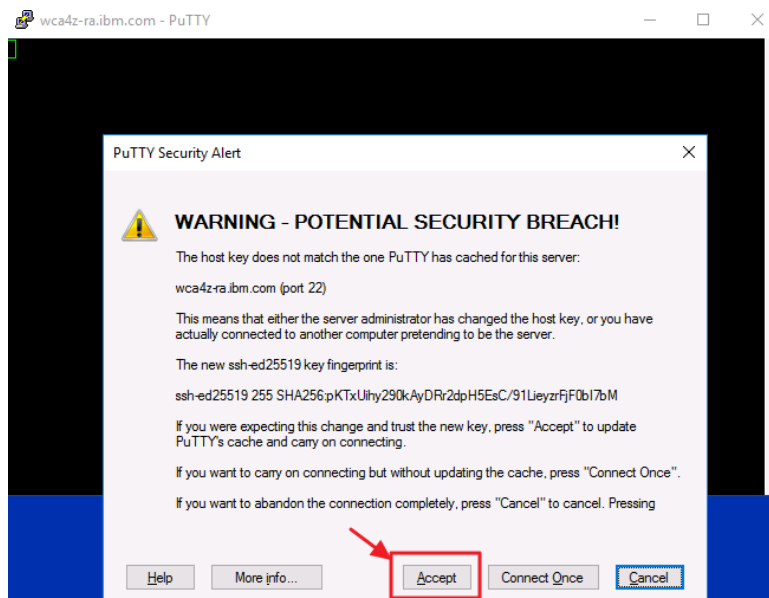
3. On RDP browser, Open Putty from taskbar.



4. select wca4z-ra and click on open.

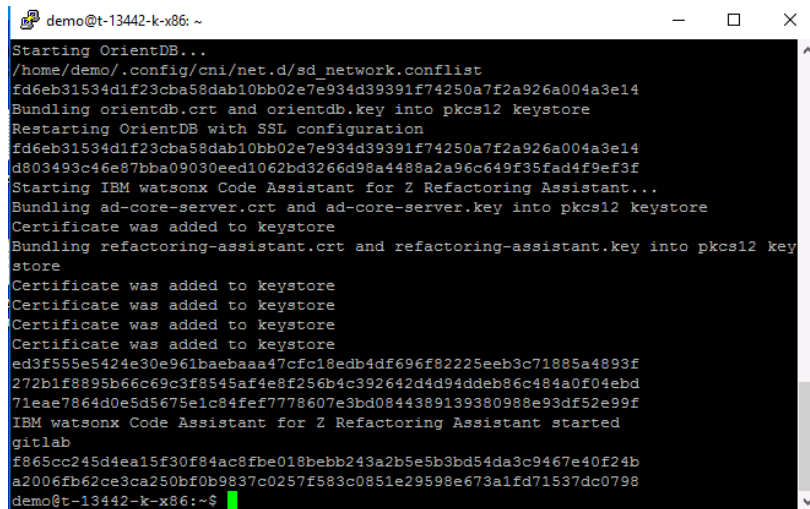


5. After clicking on Open in previous step following warning will pop-up. Select "Accept".
This will be displayed only first time.



6. Auto scripts will run on Putty to start the Refactoring Assistant.

Wait till message “IBM Watson code assistant for Z Refactoring Assistant started” is displayed on putty. After this message close putty using close button on right top.



```
demo@t-13442-k-x86: ~  
Starting OrientDB...  
/home/demo/.config/cni/net.d/sd_network.conflist  
fd6eb31534d1f23cba58dab10bb02e7e934d39391f74250a7f2a926a004a3e14  
Bundling orientdb.crt and orientdb.key into pkcs12 keystore  
Restarting OrientDB with SSL configuration  
fd6eb31534d1f23cba58dab10bb02e7e934d39391f74250a7f2a926a004a3e14  
d803493c46e87bba09030eed1062bd3266d98a4488a2a96c649f35fad4f9ef3f  
Starting IBM watsonx Code Assistant for Z Refactoring Assistant...  
Bundling ad-core-server.crt and ad-core-server.key into pkcs12 keystore  
Certificate was added to keystore  
Bundling refactoring-assistant.crt and refactoring-assistant.key into pkcs12 keystore  
Certificate was added to keystore  
Certificate was added to keystore  
Certificate was added to keystore  
Certificate was added to keystore  
ed3f555e5424e30e961baebaaa47cfc18edb4df696f82225eeb3c71885a4893f  
272b1f8895b66c69c3f8545af4e8f256b4c392642d4d94ddeb86c484a0f04ebd  
71eae7864d0e5d5675e1c84fef7778607e3bd0844389139380988e93df52e99f  
IBM watsonx Code Assistant for Z Refactoring Assistant started  
gitlab  
f865cc245d4ea15f30f84ac8f8e018bebb243a2b5e5b3bd54da3c9467e40f24b  
a2006fb62ce3ca250bf0b9837c0257f583c0851e29598e673a1fd71537dc0798  
demo@t-13442-k-x86:~$
```

3. How to Understand and Refactor your COBOL programs using IBM watsonx Code Assistant for Z Refactoring Assistant

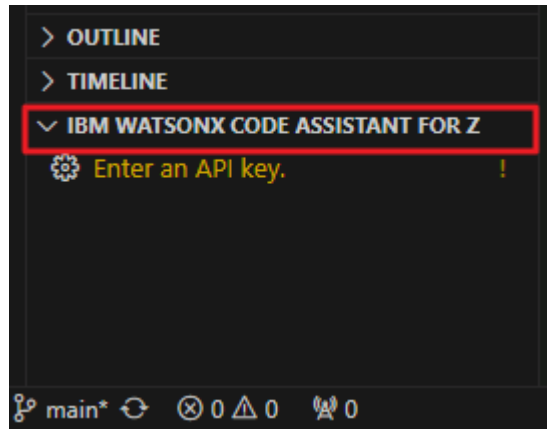
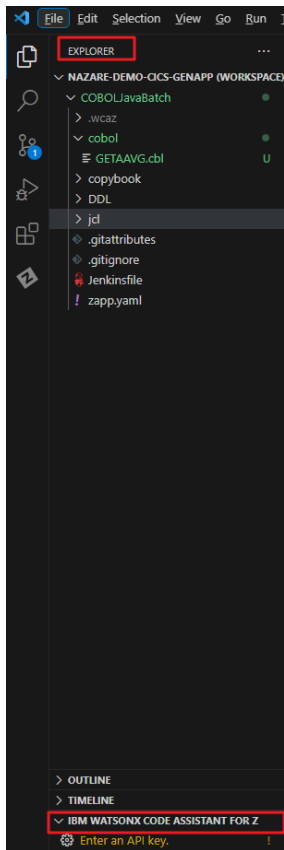
The initial installation and environment level setup will be done by the system programmer. Below initial setup steps are for the user who will be using the IBM watsonx Code Assistant for Z Refactoring Assistant in zVA environment.

Procedures to access refactoring assistant on zVA.

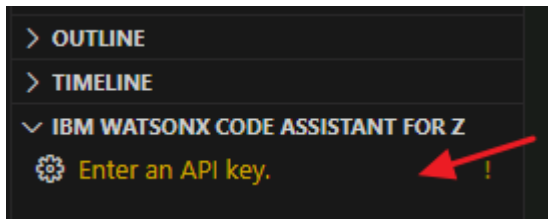
1. Open Visual Studio from desktop shortcut On RDP browser.



2. On the **Explore** menu, expand the “IBM WATSONX CODE ASSISTANT FOR Z” at the left corner of the visual studio.



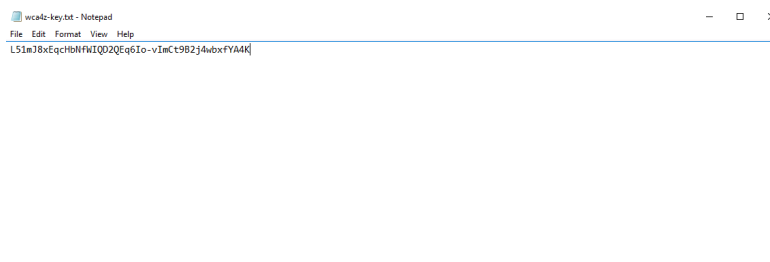
3. You can see warning “Enter an API key” in yellow colour.



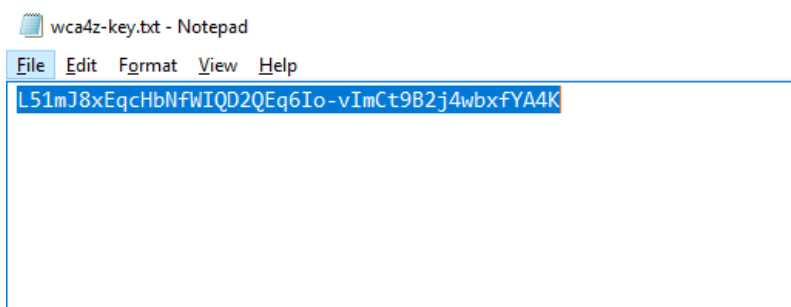
4. To get the API key, minimize all windows and go to desktop.



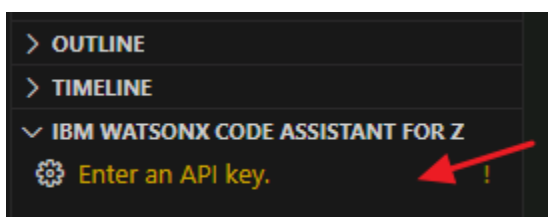
5. Click on notepad named “wca4z-key.txt” to open it.



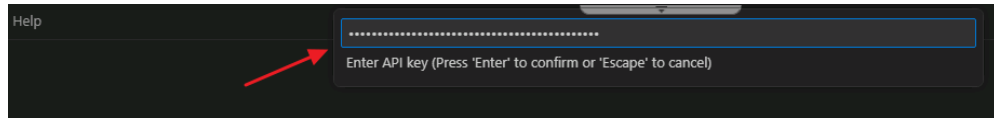
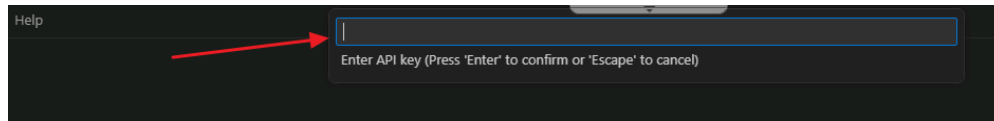
6. Copy API key from the notepad and go back to VS code.



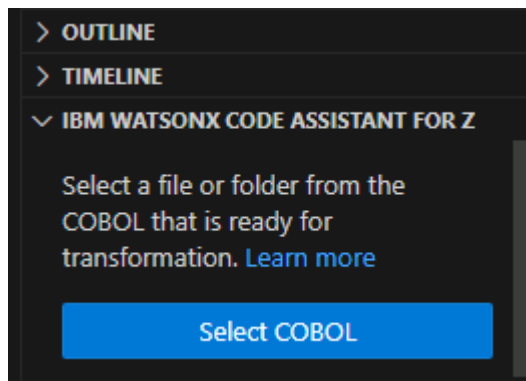
7. Click on warning to get an option “Enter an API key” besides the warning. Click on this option.



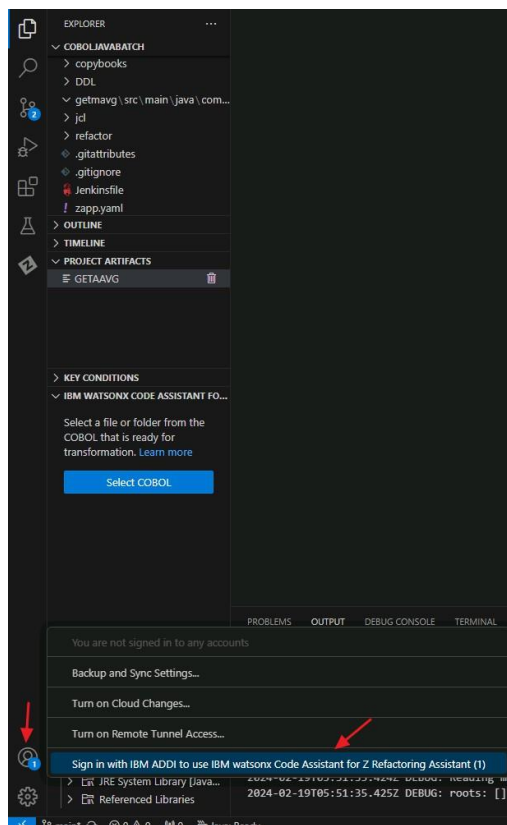
8. At the top middle an input bar will open. Give API key just copied from notepad and press enter.



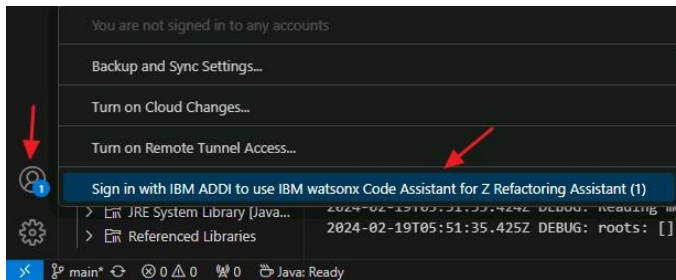
9. "Enter an API Key" warning will disappear.



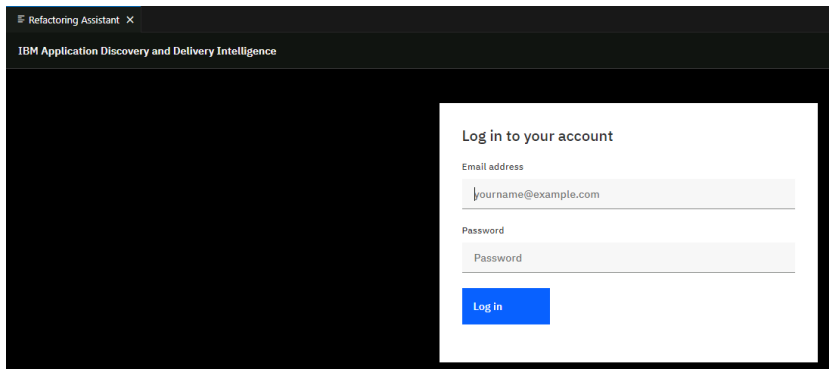
10. Click on the user icon at the bottom left of Visual Studio.



11. Click on "Sign in with IBM ADDI to use IBM watson Code Assistant for Z refactoring Assistant".



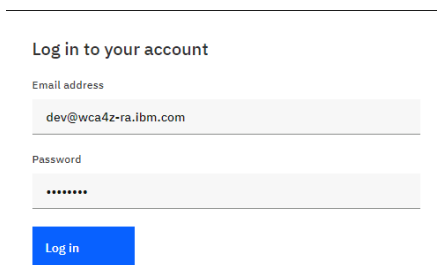
12. New tab 'Refactoring Assistant' will open for login.



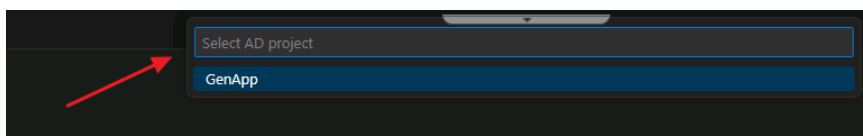
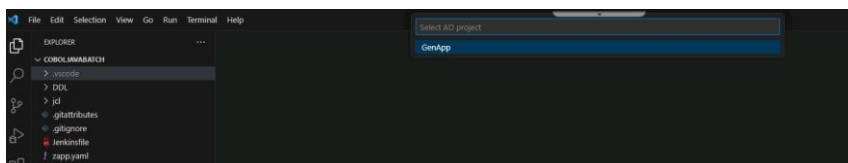
13. Select the Login credentials shown below and Click on "Log in"

ID: dev@wca4z-ra.ibm.com

Password: password



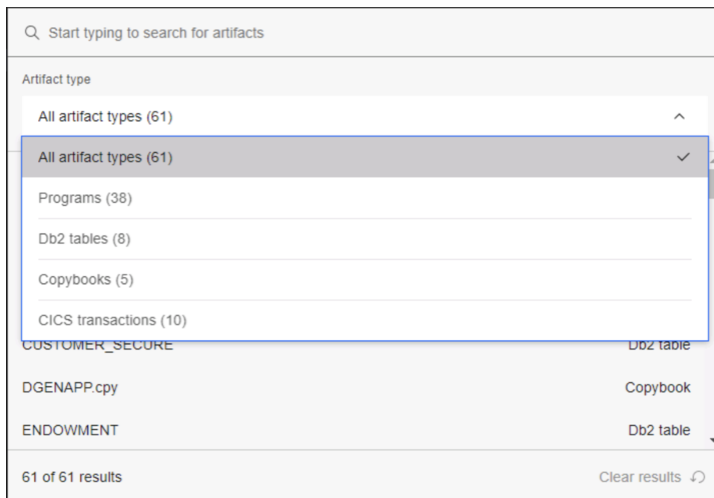
14. In the middle top, new input filed will appear, select 'GenApp' there.



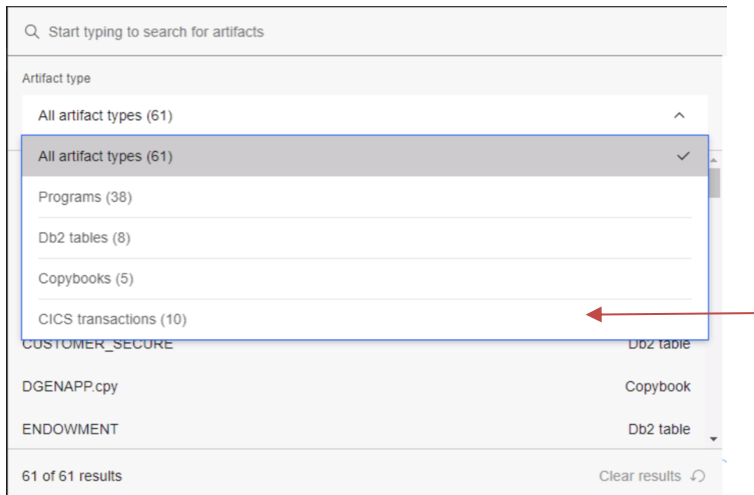
15. Click on Search bar, it will display 'all artifact types'.



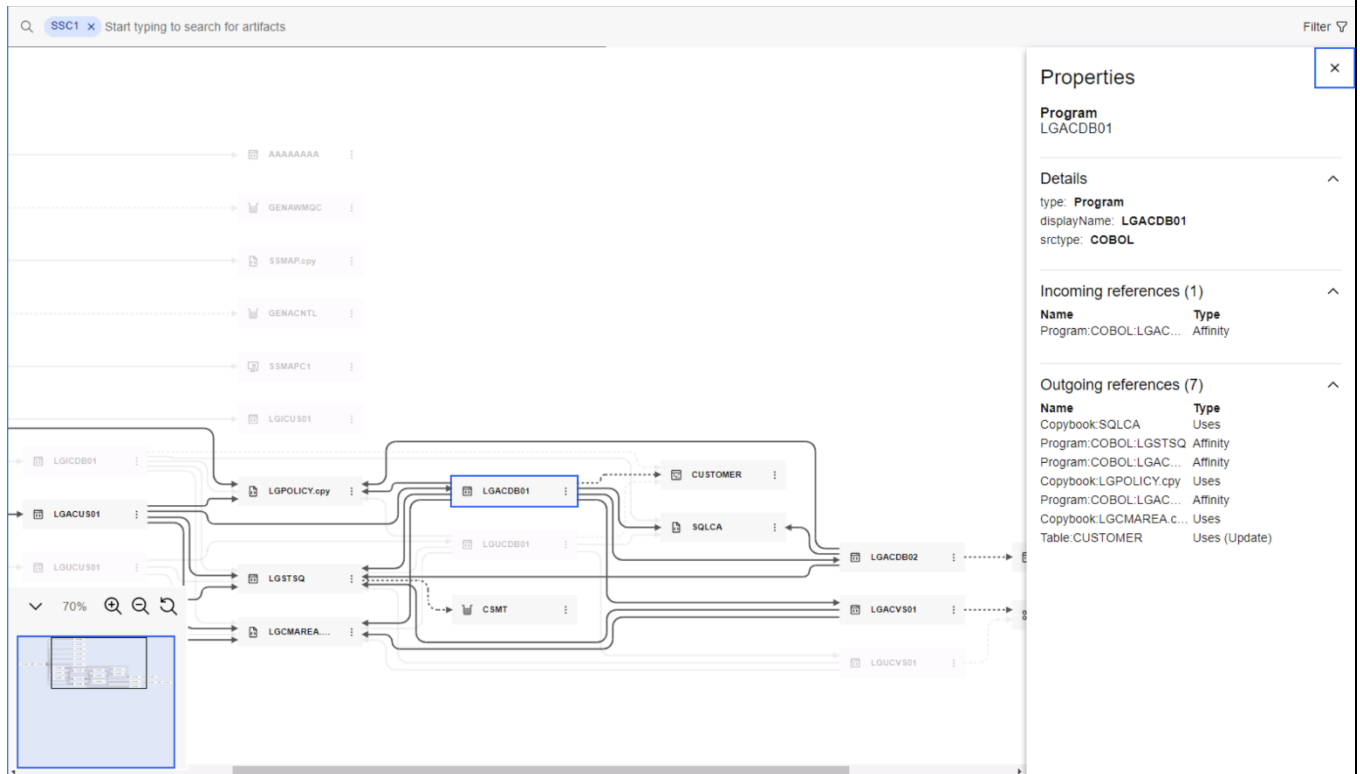
16. Click on “All artifact types” to get a dropdown with different artifact types.



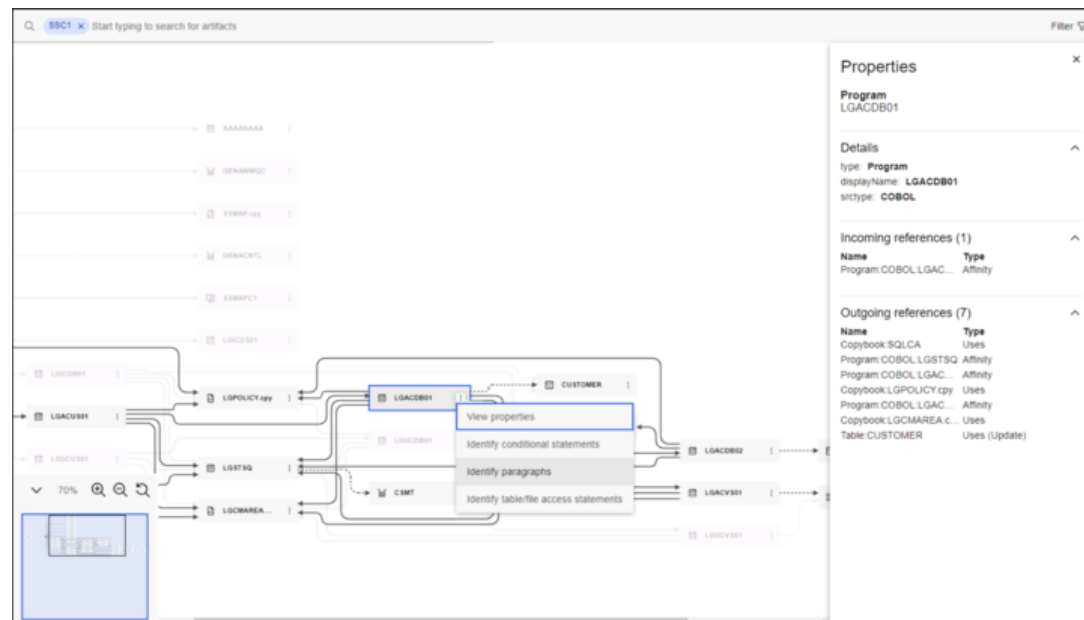
17. Select "CICS transactions".



18. From the dropdown list of transactions, click on SSC1 and press enter.



21. In the Understand phase, we checked the callgraph for SSC1 transactions and we checked the insert Customer query in the code LGACDB01.



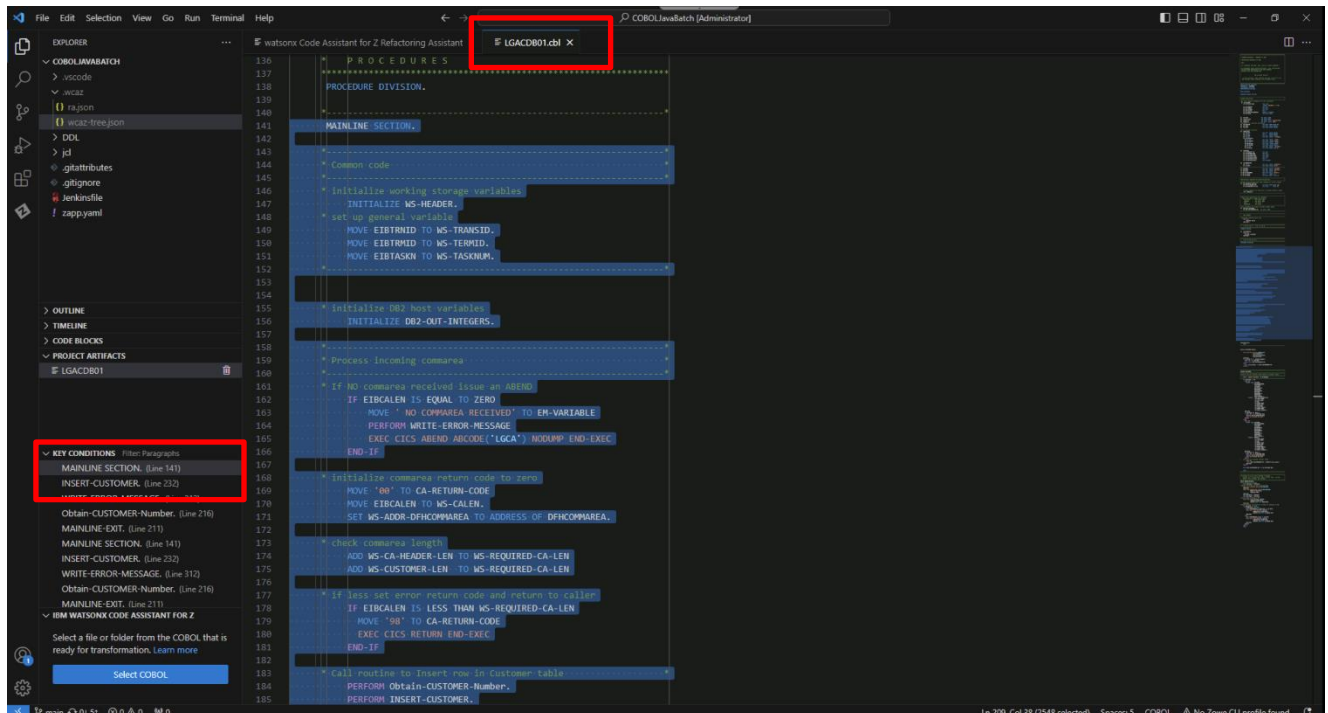
Click on the 3 dots/ellipses besides the LGACDB01 to get below options.

- View properties
- Identify conditional statements.
- Identify paragraphs.
- Identify tables/file access statements.

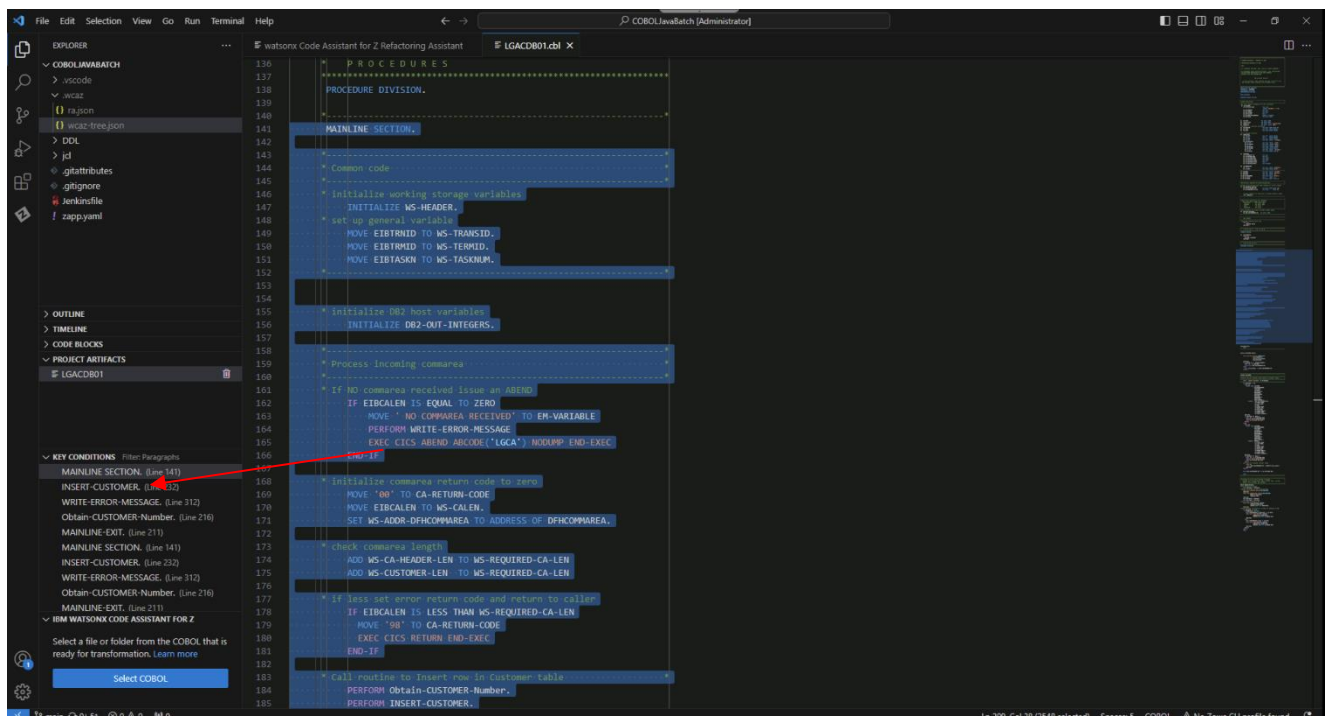
22. Click on “**Identify paragraphs**” option.

23. It will open

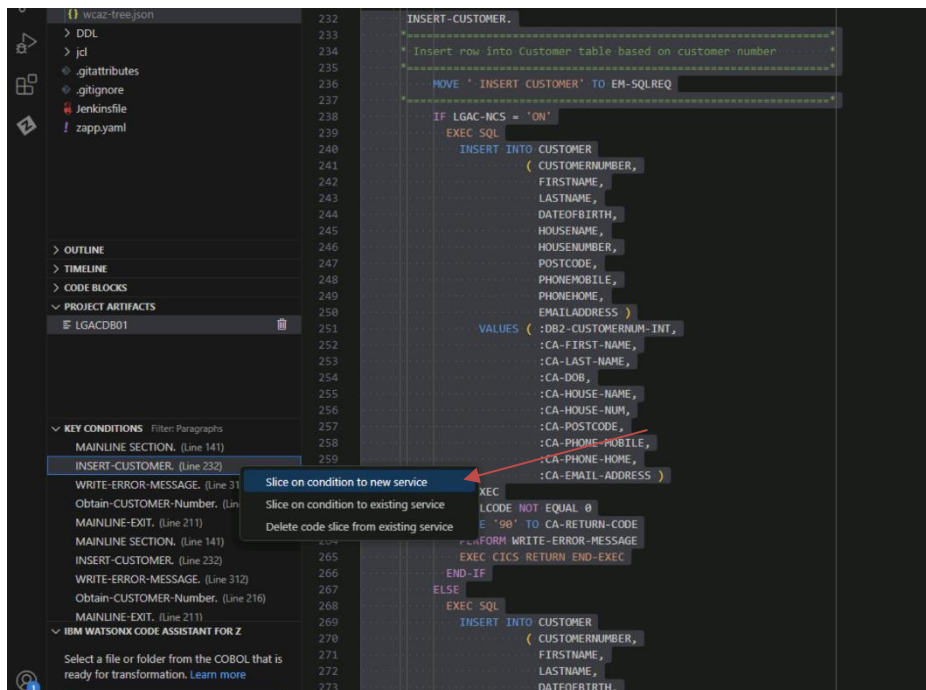
- All paragraphs in the code in the order of importance/complexity in the left side section
- Code is displayed on right side section.



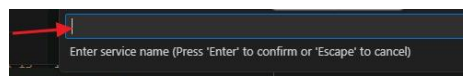
24. As we are checking Insert Customer function, Click on **INSERT-CUSTOMER** paragraph under Key conditions to see the paragraph in the code.



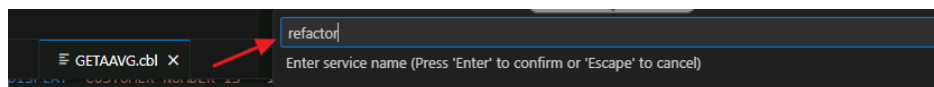
25. Right Click on the paragraph name **INSERT-CUSTOMER** appearing in the code you will get option "Slice on condition to new service". Click on this option.



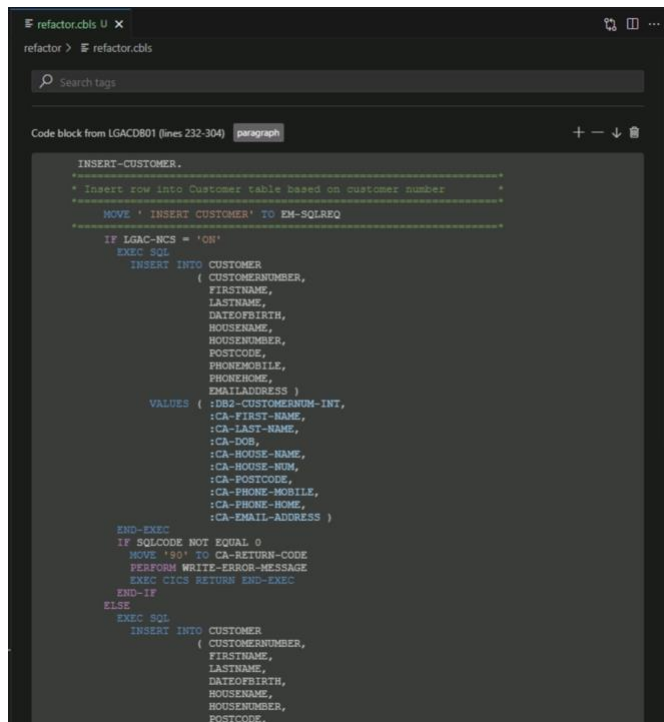
26. New input field will appear at the middle top.



27. Give the name 'refactor', and press enter.



28. The code will be sliced into new service 'refactor' opened on right side section. Close this 'refactor' tab.



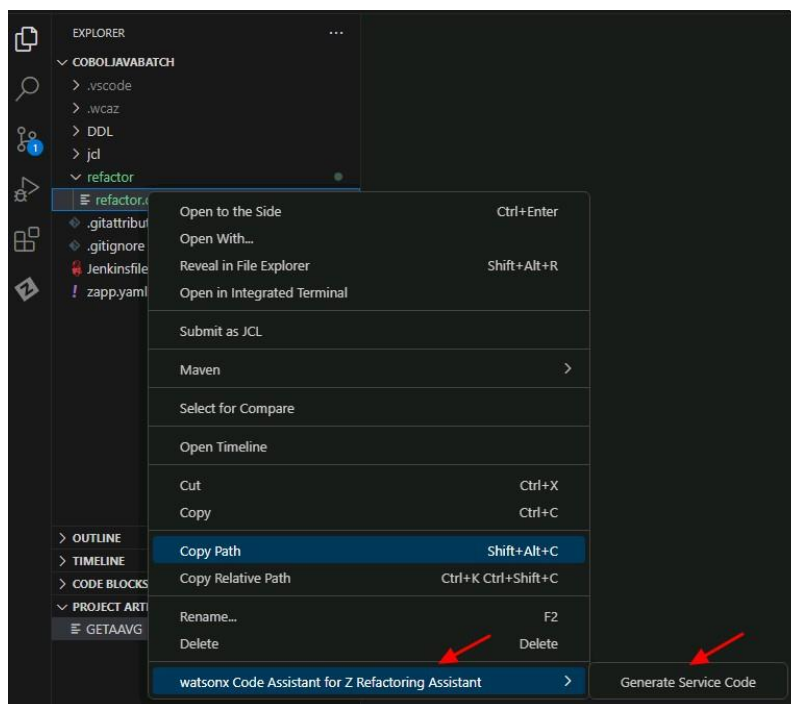
```
refactor.cbis U X
refactor > refactor.cbis

Search tags

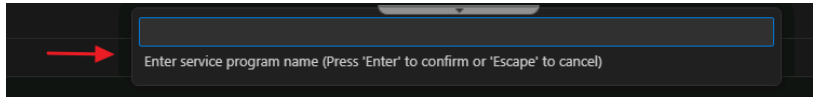
Code block from LGACD801 (lines 232-304) paragraph

INSERT-CUSTOMER.
-----
* Insert row into Customer table based on customer number
*
MOVE 'INSERT CUSTOMER' TO EM-SQLREQ
-----
IF LGAC-NCS = 'CH'
  EXEC SQL
    INSERT INTO CUSTOMER
      ( CUSTOMERNUMBER,
        FIRSTNAME,
        LASTNAME,
        DATEOFBIRTH,
        HOUSENAME,
        HOUSENUMBER,
        POSTCODE,
        PHONEMOBILE,
        PHONENOME,
        EMAILADDRESS )
    VALUES ( :DB2-CUSTOMERNUM-INT,
              :CA-FIRST-NAME,
              :CA-LAST-NAME,
              :CA-DOB,
              :CA-HOUSE-NAME,
              :CA-HOUSE-NUM,
              :CA-POSTCODE,
              :CA-PHONE-MOBILE,
              :CA-PHONE-HOME,
              :CA-EMAIL-ADDRESS )
  END-EXEC
  IF SQLCODE NOT EQUAL 0
    MOVE '99' TO CA-RETURN-CODE
    PERFORM WRITE-ERROR-MESSAGE
    EXEC CICS RETURN END-EXEC
  END-IF
ELSE
  EXEC SQL
    INSERT INTO CUSTOMER
      ( CUSTOMERNUMBER,
        FIRSTNAME,
        LASTNAME,
        DATEOFBIRTH,
        HOUSENAME,
        HOUSENUMBER,
        POSTCODE,
```

29. Write Click on the **refactor.cbis** from left section and go to 'Refactoring Assistant', then click on 'Generate Service Code'.



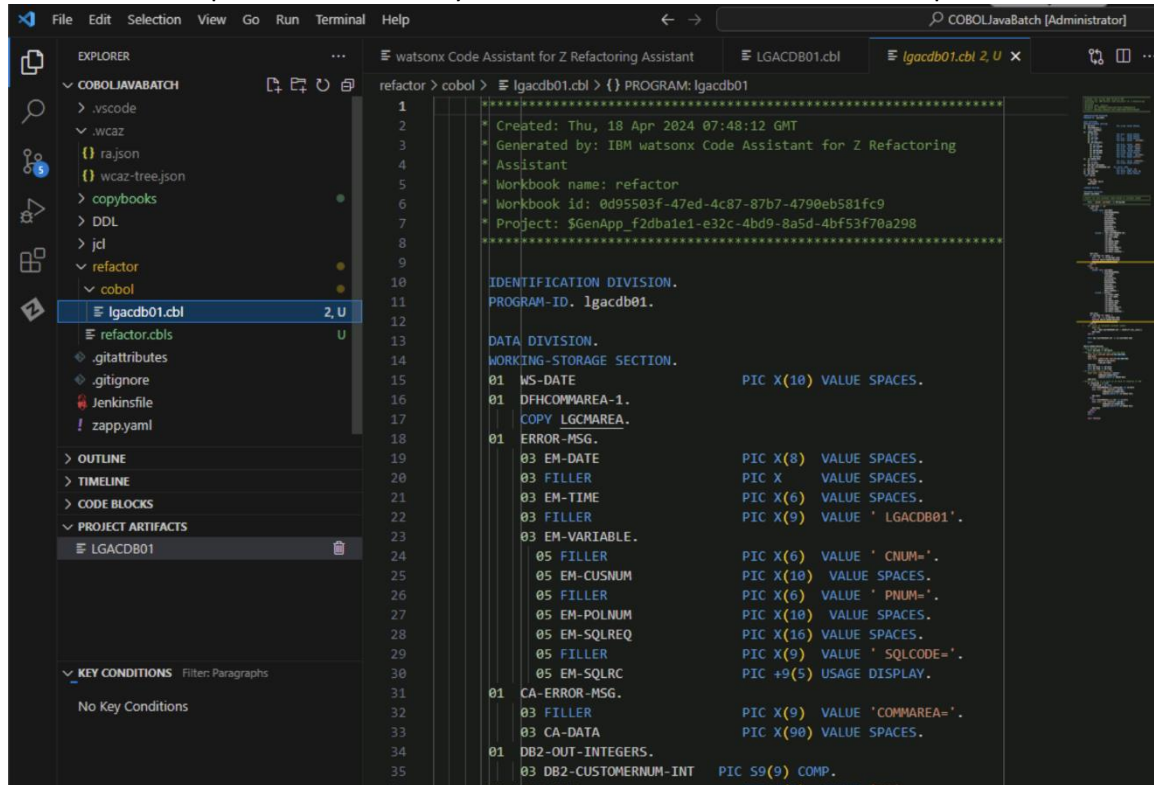
30. New input filed will appear in the middle top.



31. Change name to **LGAOCD01** and press **Enter**.

32. New tab will open with refactored code '**LGAOCD01.cbl**'.

If not opened automatically, double click on '**LGAOCD01.cbl**' to open it.

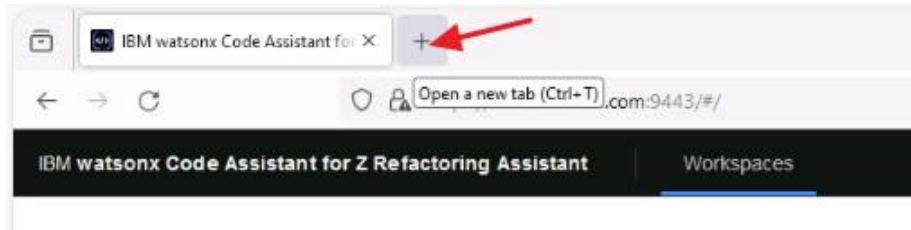


In this **Understand** and **Refactor** phase,

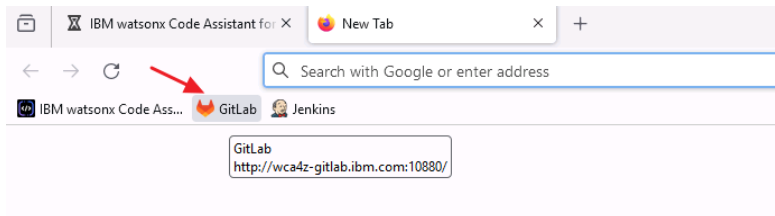
1. We selected SSC1 transaction to understand the flow and related components.
2. We checked the data access program LGACDB01 to insert customer in data table Customer.
3. We used IBM watsonx Code Assistant for Z Refactoring Assistant for refactoring the insert customer functionality by slicing code LGACDB01 into service code.
4. This service code will be used in next Transform phase.

4. How to execute the Transform Phase using VS Code on WCA4Z and auto trigger Jenkins process using GitLab.

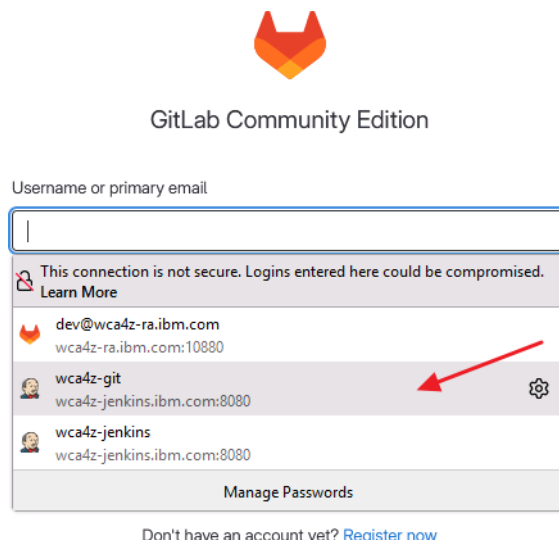
1. Continue in the VS code window from previous step.
2. Go to web browser and add click on new tab.



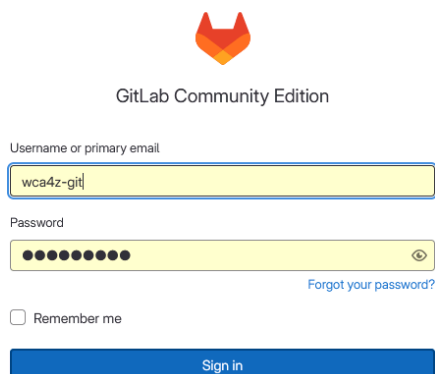
3. Click on **"GitLab"** from the favourite bar.



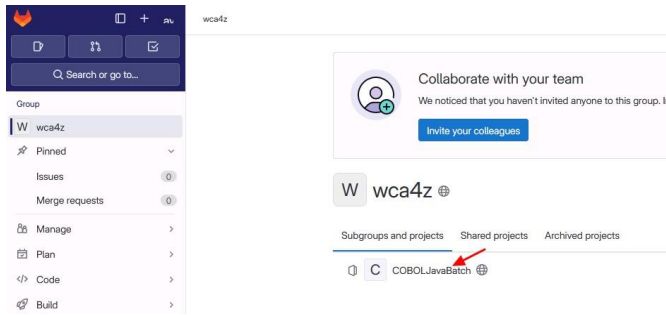
4. GitLab page will open and ask for login credentials. From list of credentials, select GitLab credentials **'wca4z-git'** as shown below.



5. GitLab Id and password will be prepopulated. Click on **'Sign In'**.



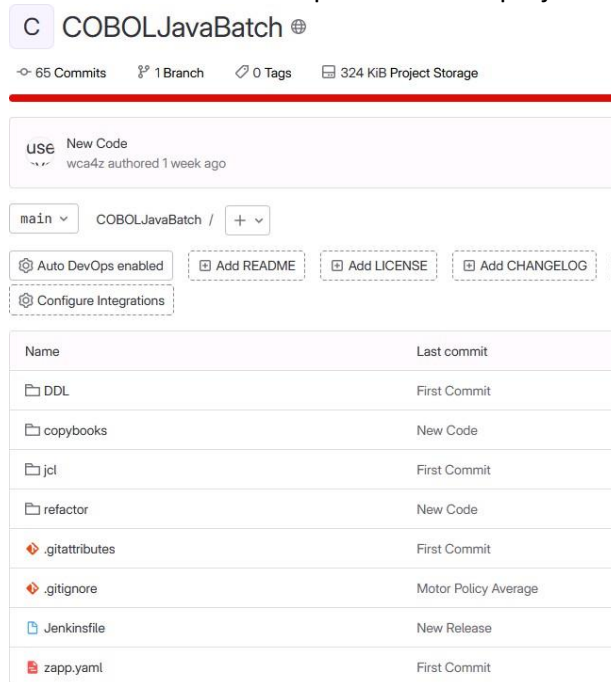
6. After successful login GitLab Projects page will open. Click on Project name **'wca4z/COBOLJavaBatch'** to open it.



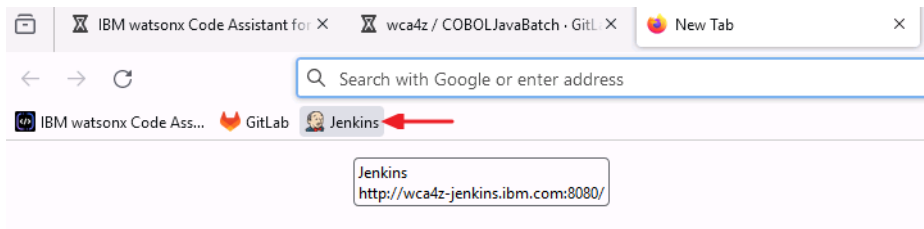
7. Once project is opened, all the artifacts in the projects are listed.

Please note that

- a. There are no COBOL components in the project till now.



8. Open a new tab in the browser and click on 'Jenkins' link from favourite bar.



9. Jenkins login page will open, Select Jenkins credentials – 'wca4z-jenkins' from the list.

Sign in to Jenkins

Username

This connection is not secure. Logins entered here could be compromised.
[Learn More](#)

- wca4z-git
From this website
- wca4z-jenkins
From this website
- dev@wca4z-ra.ibm.com
wca4z-ra.ibm.com:10880

Manage Passwords

Sign in

10. Click on 'Sign In'.
Sign in to Jenkins

Username

Password

☐ Keep me signed in

Sign in

11. After successful login, Jenkins Dashboard page will open with
'ADDIBuildWCA4ZProject' displayed.

Jenkins

Dashboard >

+ New Item

Only use for wca4z Builds.

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

S	W	Name ↓
...	☀	ADDIBuildWCA4ZProject
🕒	☀	ADDIServerConfigure

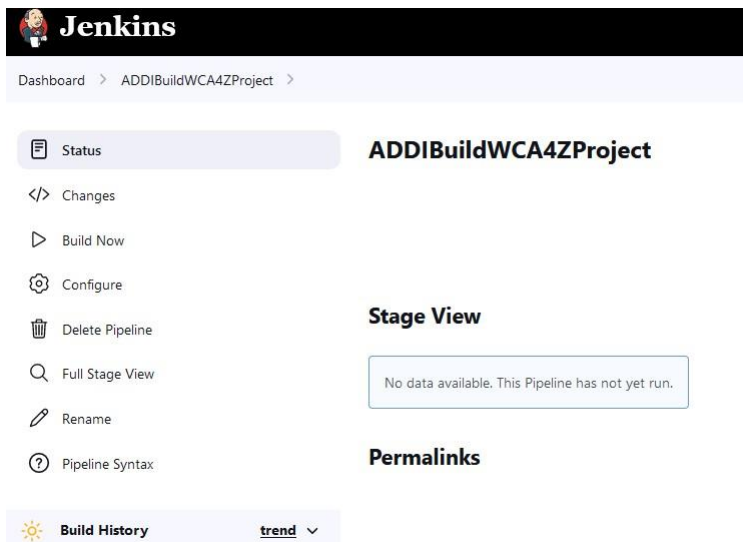
12. Click on the 'ADDIBuildWCA4ZProject' name to see the list of jobs.

Name ↓

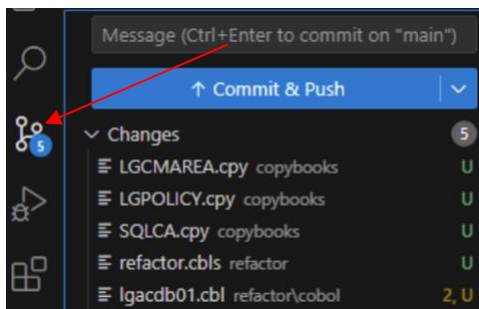
ADDIBuildWCA4ZProject

ADDIServerConfigure

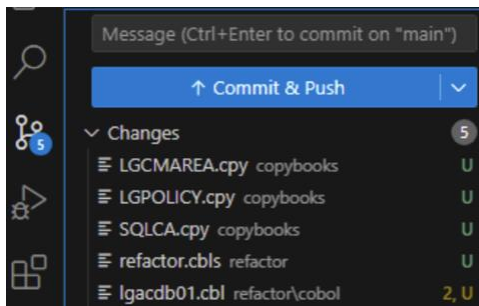
13. No previous jobs are present.



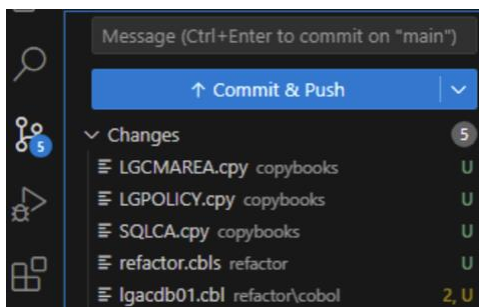
14. Minimize the browser and go to VS code. Click on the '**SOURCE CONTROL**' icon from the left side icons as shown below.



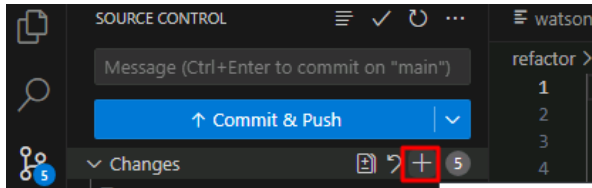
15. SOURCE CONTROL tab will open which will have the new LGACDB01 cobol file, and the copybooks.



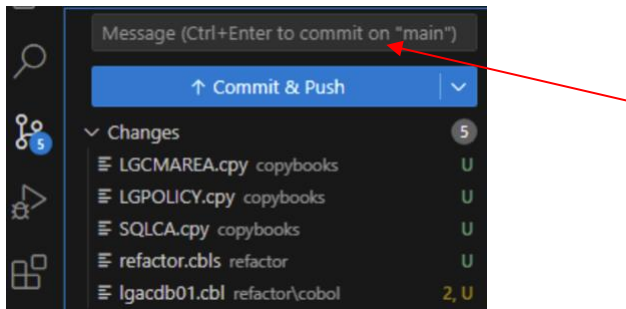
16. Click on the '+' sign in the **Changes** row to ensure all the cobol programs and the associated copybooks are staged.



17. Changes will move to **'Staged Changes'**.

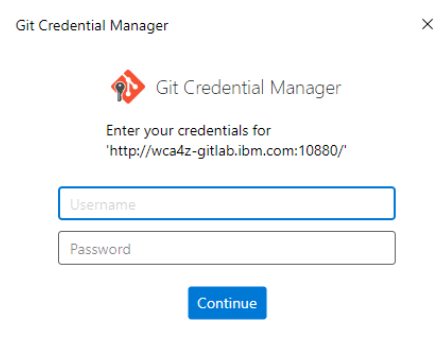


18. Add a commit message for the changes as shown. E.g., 'Adjustments after Refactoring in LGACDB01.'



19. Click on **'Commit & Push'**.

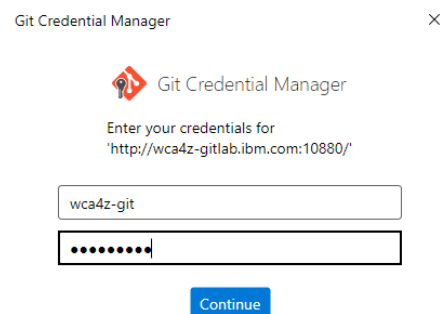
20. New Pop-up will open asking for GitLab credentials.



21. Give Gitlab Credentials and press on **'Continue'**.

ID: wca4z-git

Password : P@assw0rd



22. Commit & Push will be done successfully.

23. Go to browser to already opened Jenkins page and refresh it.

24. New job is triggered after commit & Push changes. Wait till all stages of job are completed.

The screenshot shows the Jenkins interface for the project 'ADDIBuildWCA4ZProject'. The 'Stage View' section displays a single stage named 'Declarative: Checkout SCM' with an average stage time of 3s. The build history shows a build on Mar 01 at 11:44 with 'No Changes'.

Stage	Average stage times
Declarative: Checkout SCM	3s

Build History: Mar 01 11:44 No Changes

25. Click on the Job #2 from left side list.

The screenshot shows the Jenkins interface for the project 'ADDIBuildWCA4ZProject'. The 'Stage View' section displays a table of stages with their durations. The build history shows a build on Mar 01 at 11:44 with 'No Changes'.

Stage	Average stage times
Declarative: Checkout SCM	7s
Git Clone	4s
WCA4Z Controlflow Generator	1min 4s
Archive	1s
Declarative: Post Actions	381ms

Build History: Mar 01 11:44 No Changes

26. Job will be opened. Click on **Console Output** from left side list.

Dashboard > ADDIBuildWCA4ZProject > #2

Status

<> Changes

Console Output

Edit Build Information

Delete build '#2'

Polling Log

Git Build Data

Git Build Data

Restart from Stage

Build #2 (Mar. 1, 2024, 5:44:16 a.m.)

Started by GitLab push by wca4z

Build Artifacts

BatchMakeStatusFile_01_Mar_2024_05_45_05.txt	166 B	view
JavaDb2Batch_01_Mar_2024_05_45_05.txt	497 B	view
runControlFlowGenerator.log	8.06 KB	view
UpdateInBackgroundLog_01_Mar_2024_05_44_48.txt	900 B	view

Triggered by GitLab Webhook

Revision: 5450f60cc7894e3e7ce985b6a23381ea65082d65
Repository: ssh://git@wca4z-gitlab.ibm.com:10022/wca4z/COBOLJavaBatch.git

• origin/main

27. Job log will open for the job.

Console Output

```
Started by GitLab push by wca4z
Obtained Jenkinsfile from git ssh://git@wca4z-gitlab.ibm.com:10022/wca4z/COBOLJavaBatch.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on ADDI_Server in C:\Jenkins_ROOT\workspace\ADDIBuildWCA4ZProject
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential GitLabID
Fetching changes from the remote Git repository
> git.exe rev-parse --resolve-git-dir C:\Jenkins_ROOT\workspace\ADDIBuildWCA4ZProject\.git # timeout=10
Checking out Revision 5450f60cc7894e3e7ce985b6a23381ea65082d65 (origin/main)
```

28. Scroll down the console output using scroll bar at the right side to view complete logs of the job.

```
[Pipeline] echo
===== executed successfully=====
[Pipeline] updatedGitLabCommitStatus
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (wca4z ControlFlow Generator)
[Pipeline] script
[Pipeline] {
[Pipeline] bat

C:\Jenkins_ROOT\workspace\ADDIBuildWCA4ZProject\c:\data\scripts\JBatch.bat
Read configuration:
ADDI_PROJECT_NAME=JavaDb2Batch
ADDI_PROJECTS_FOLDER=C:\IBM AD\VaInframe Projects
PROJECT_GITHUB_CLONE_FOLDER=C:\data\GIT_REPOS
DDL_FOLDER=C:\data\GIT_REPOS\DDL
ADDI_PROJECT_FOLDER=C:\IBM AD\VaInframe Projects\JavaDb2Batch
Using DDL file: C:\data\GIT_REPOS\DDL\vb2cre.jcl

Checking if IBM Application Discovery Configuration Service is running
STATE : 4 RUNNING

Synchronize local folders with git (git pull)
From http://wca4z-gitlab.ibm.com:10022/wca4z/COBOLJavaBatch
* branch HEAD -> FETCH_HEAD
Already up to date.

Synchronize ADDI project with local folder (IBMApplicationDiscoveryBuildClient /m1 JavaDb2Batch)
Project synchronization log: "C:\IBM AD\VaInframe Projects\JavaDb2Batch\UpdateInBackgroundLog_15_Jan_2024_22_33_34.txt"

Incremental build of ADDI project (IBMApplicationDiscoveryBuildClient /m1 JavaDb2Batch)
Project build log: "C:\IBM AD\VaInframe Projects\JavaDb2Batch\JavaDb2Batch_15_Jan_2024_22_33_34.txt"
Found updated file: "GETAWG.cbl"
Resolved path to file: "C:\data\GIT_REPOS\cobol\GETAWG.cbl"
Resolved program name: "GETAWG"

Run the process to populate additional metadata for WCA4Z from ADDI and DDL
Program=GETAWG
Generating PrimaryKeys.csv and ForeignKeys.csv
Importing PrimaryKeys.csv and ForeignKeys.csv into the database
```

```

Synchronize ADO1 project with local folder (IBMApplicationDiscoveryBuildClient /uml JavaDb2Batch)
Project synchronization log: "C:\IBM AD\Mainframe Projects\JavaDb2Batch\UpdateInBackgroundLog_15_Jan_2024_22_33_34.txt"

Incremental build of ADO1 project (IBMApplicationDiscoveryBuildClient /ml JavaDb2Batch)
Project build log: "C:\IBM AD\Mainframe Projects\JavaDb2Batch\JavaDb2Batch_15_Jan_2024_22_33_54.txt"
Found updated file: "GETAAWG.chl"
Resolved path to file: "C:\data\GIT_REPOS\cobol\GETAAWG.chl"
Resolved program name: "GETAAWG"

Run the process to populate additional metadata for WCA4Z from ADO1 and DDL
Program=GETAAWG
Generating PrimaryKeys.csv and ForeignKeys.csv
Importing PrimaryKeys.csv and ForeignKeys.csv into the database
Done.
[Pipeline] }
[Pipeline] // script
[Pipeline] updateGitLabCommitStatus
Post stage
[Pipeline] echo
=====Layers=====
[Pipeline] echo
-----A executed successfully-----
[Pipeline] updateGitLabCommitStatus
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
-----pipeline executed successfully -----
[Pipeline] updateGitLabCommitStatus
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withenv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

29. On the browser go to GitLab page and refresh it.

As shown below

- Latest commit message can be seen with green tick.
- New **refactor** folder is visible now.

wca4z / COBOLJavaBatch

USE refactored code
wca4z authored 1 minute ago

471c85ca

main COBOLJavaBatch / +

History Find file Edit Code

Name	Last commit	Last update
DDL	First Commit	3 months ago
copybooks	New Code	4 hours ago
jcl	First Commit	3 months ago
refactor	refactored code	47 minutes ago
.gitattributes	First Commit	3 months ago

30. Click on **refactor** folder.

wca4z / COBOLJavaBatch

USE refactored code
wca4z authored 1 minute ago

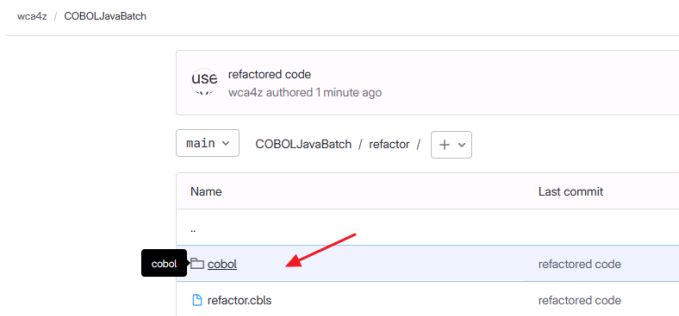
471c85ca

main COBOLJavaBatch / +

History Find file Edit Code

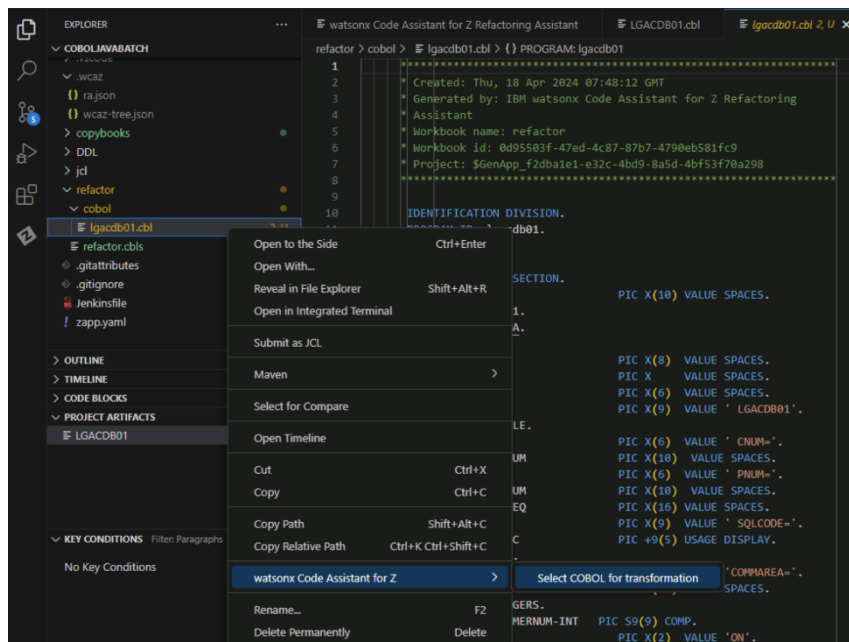
Name	Last commit	Last update
DDL	First Commit	3 months ago
copybooks	New Code	4 hours ago
jcl	First Commit	3 months ago
refactor	refactored code	47 minutes ago
.gitattributes	First Commit	3 months ago

31. Click on **cobol** folder.

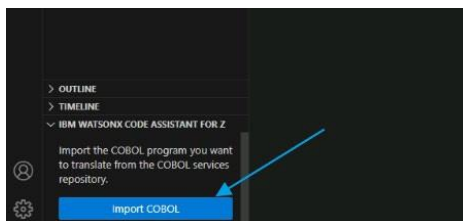


32. New code LGACDB01 will be present with the commit message.

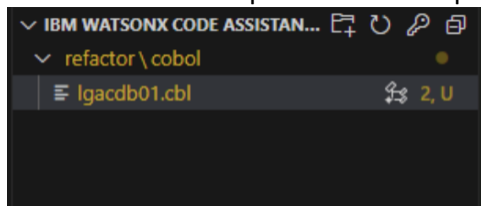
33. Go back to VS Code and right click on name of the code LGACDB01 and go to **“watsonx Code Assistant for Z”** then click on option **“Select COBOL for Translation”**.



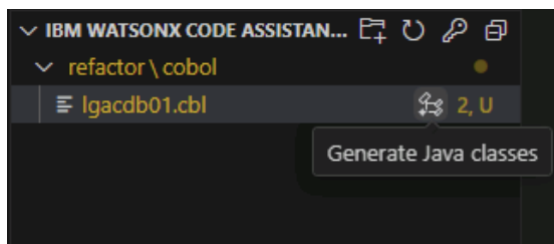
34. Option Step: This can also be done from the blue button at the left bottom “Import COBOL program” and select browse the code location and select LGACDB01 program.



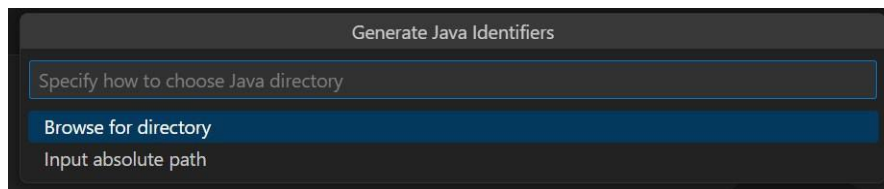
35. This will import the COBOL program into “IBM WATSONX CODE ASSISTANT FOR Z”.



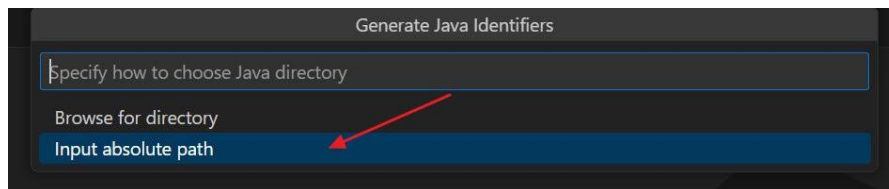
36. Click on icon next to ‘cobol\lgacdb01.cbl’ to Generate Java Classes.



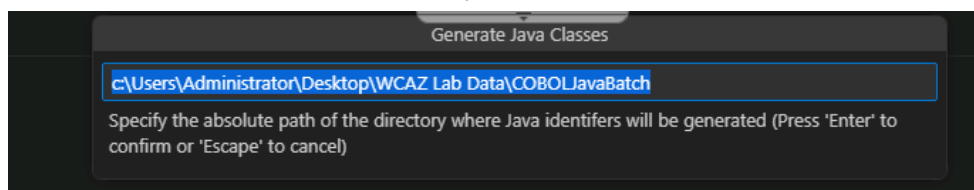
37. This will open new input bar in the top middle.



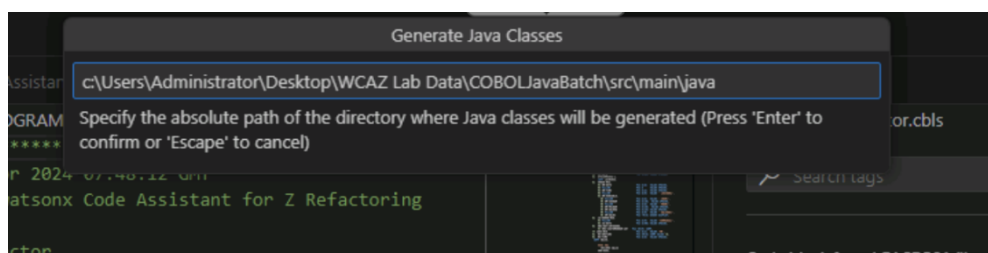
38. Click on “Input absolute path” option.



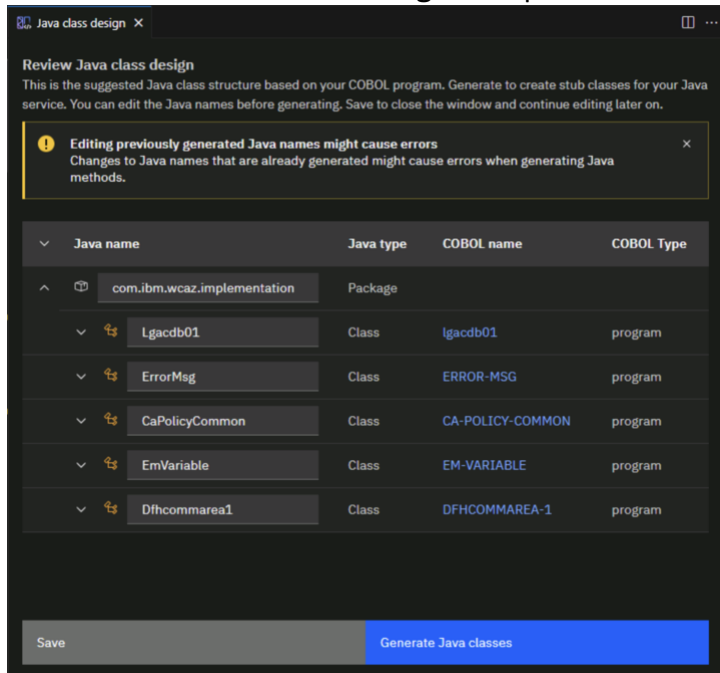
39. Press end to reach end of the default path.



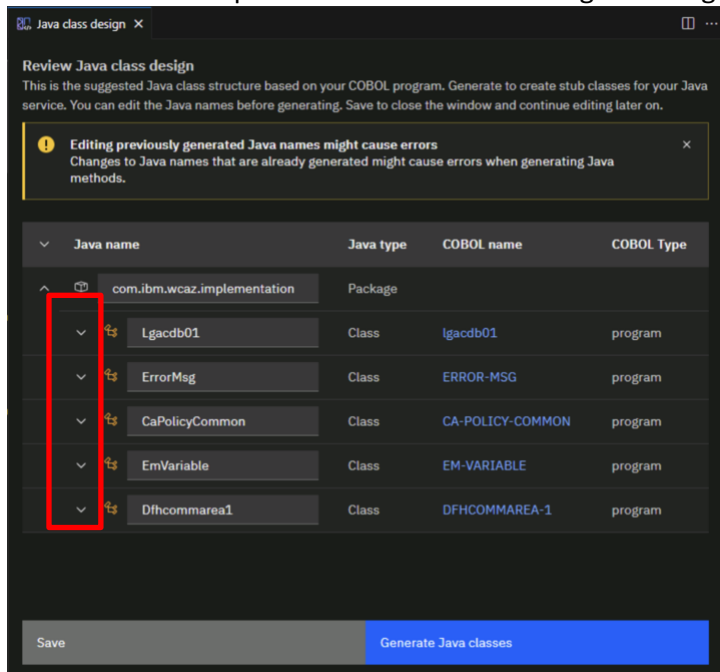
40. At the end, type path as \src\main\java and press enter.



41. New tab **Java Class design** will open.



42. Click to expand different artifacts using following buttons.



43. At the right bottom of 'Java Class design' tab, click on "Generate Java classes".

Java class design

Review Java class design

This is the suggested Java class structure based on your COBOL program. Generate to create stub classes for your Java service. You can edit the Java names before generating. Save to close the window and continue editing later on.

!


Editing previously generated Java names might cause errors
Changes to Java names that are already generated might cause errors when generating Java methods.

×

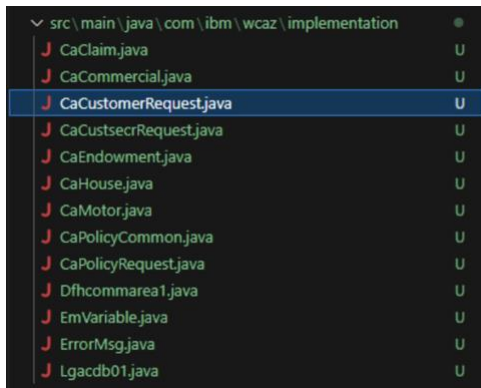
Java name	Java type	COBOL name	COBOL Type
com.ibm.wcaz.implementation	Package		
Lgacdb01	Class	lgacdb01	program
ErrorMsg	Class	ERROR-MSG	program
CaPolicyCommon	Class	CA-POLICY-COMMON	program
EmVariable	Class	EM-VARIABLE	program
Dfhcommarea1	Class	DFHCOMMAREA-1	program

Save

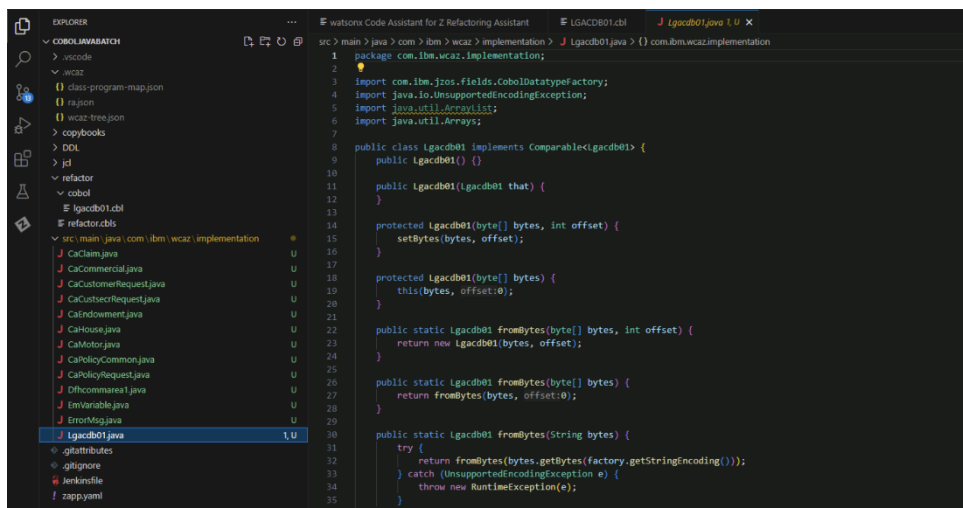
Generate Java classes



44. In the explorer, check the folder (src\main\java\com\ibm\wcaz) for generated Java classes.



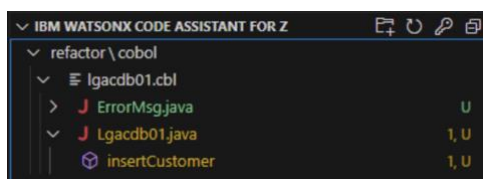
45. Click to open the “Lgacdb01.java”.



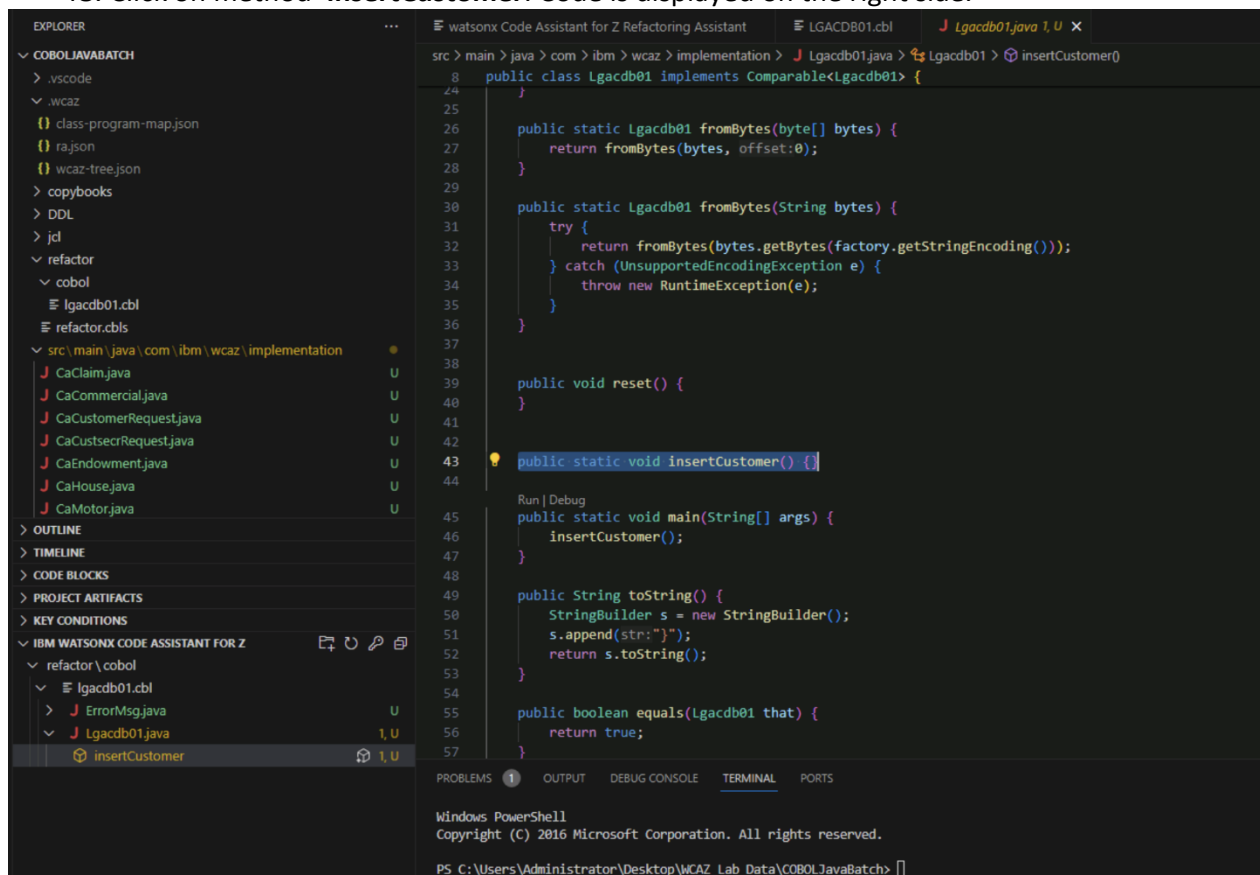
46. The ‘**JAVA PROJECTS**’ will appear at the left bottom of the Explorer, and it will start building the java project. Java projects will be generated.



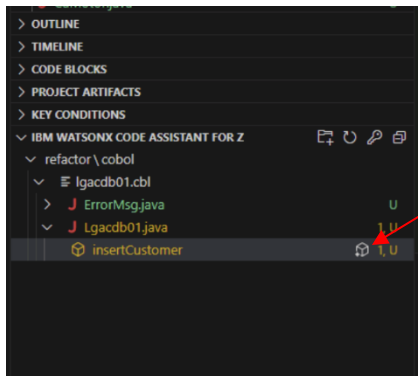
47. You can find the Java classes and method names under 'IBM WATSONX CODE ASSISTANT FOR Z' -> 'refactor\cobol' folder



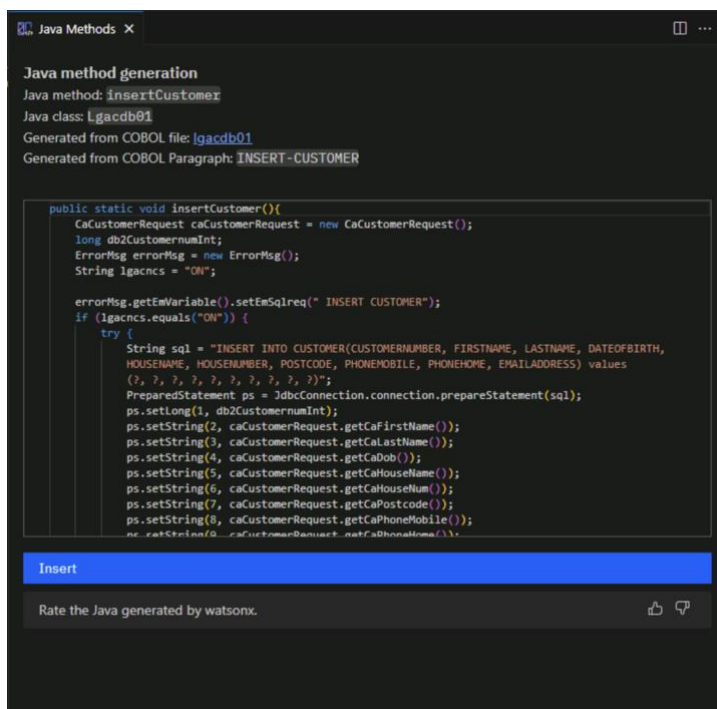
48. Click on method 'insertCustomer'. Code is displayed on the right side.



49. Click on the icon next to "insertCustomer" in the IBM Watsonx Code Assistant for Z dropdown.



50. Once clicked on 'Generate Java method', Java method gets generated on the right, click on insert

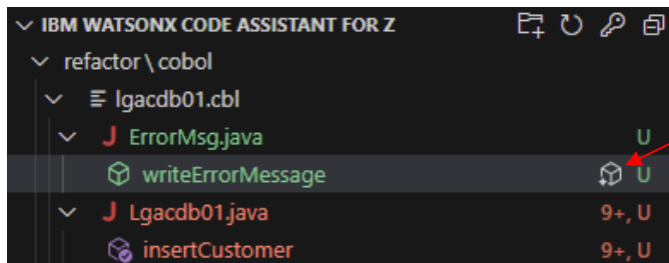


51. Generated java method for 'insertCustomer' will get inserted in the java class.

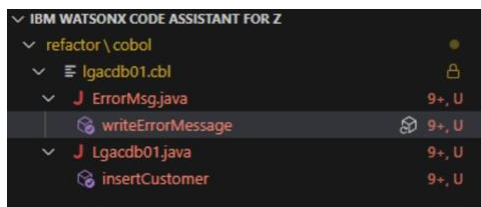
```
public static void insertCustomer(){
    CaCustomerRequest caCustomerRequest = new CaCustomerRequest();
    long db2CustomerNumInt;
    ErrorMessage errorMsg = new ErrorMessage();
    String lgacnscs = "ON";

    errorMsg.getMessage().setSqlReq("INSERT CUSTOMER");
    if (lgacnscs.equals("ON")) {
        try {
            String sql = "INSERT INTO CUSTOMER(CUSTOMERNUMBER, FIRSTNAME, LASTNAME, DATEOFBIRTH, HOUSENAME, HOUSENUMBER, POSTCODE, PHONEMOBILE, PHONEHOME, EMAILADDRESS) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = JdbcConnection.getConnection().prepareStatement(sql);
            ps.setLong(1, db2CustomerNumInt);
            ps.setString(2, caCustomerRequest.getCaFirstName());
            ps.setString(3, caCustomerRequest.getCaLastName());
            ps.setString(4, caCustomerRequest.getCaDob());
            ps.setString(5, caCustomerRequest.getCaHouseName());
            ps.setString(6, caCustomerRequest.getCaHouseNum());
            ps.setString(7, caCustomerRequest.getCaPostcode());
            ps.setString(8, caCustomerRequest.getCaPhoneMobile());
            ps.setString(9, caCustomerRequest.getCaPhoneHome());
            ps.setString(10, caCustomerRequest.getCaEmailAddress());
            ps.executeUpdate();
            ps.close();
        } catch (SQLException exception) {
            caCustomerRequest.setCaReturnCode(Integer.parseInt("99"));
            errorMsg.writeErrorMessage(exception.getMessage());
            return;
        }
    }
}
```

52. Repeat this process for "writeErrorMessage" function.



53. All the methods are now tick marked that means it is not empty.



In this **Transform** phase,

1. We used the workbook LGACDB01 from Refactor phase for insert customer functionality.
2. We generated Java classes and Java methods.

We executed steps for **Understand**, **Refactor** and **Transform** phases to create java classes and methods for **Insert Customer** functionality in **LGACDB01** programs.