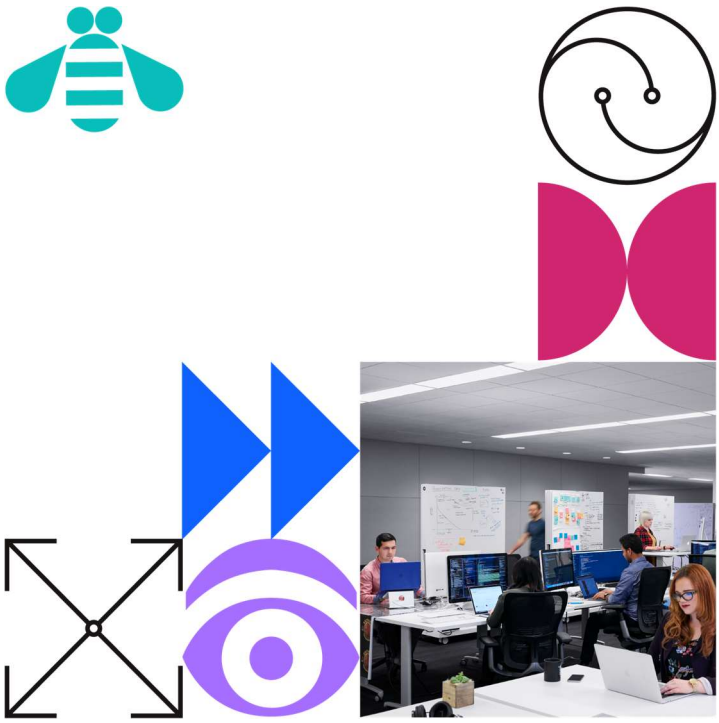
DevOPs ADDI workshop

Revamp Your Apps:
Master Modernization with IBM ADDI

ADDI Lab: Identify API candidates from your code



David Hawreluk

IBM, DevOps Technical Brand Specialist

Contents

Workshop Introduction - Identify API candidates and refactoring effort from your code

Lab Introduction

Our scenario is based on two sample applications that have been slightly modified for the purpose of the exercise: GenApp and MortgageApp. Note that there may be other applications/projects shown.

GenApp is a COBOL CICS insurance application that simulates transactions made by an insurance company to create and manage its customers and insurance policies.

Mortgage Application is a COBOL CICS application that calculates monthly mortgage payment.

Goal: We want to assess the work to be done to expose an API that reads Customer information:

- Is there a program ready to be exposed? (hint: no , but with ADDI you will find the programs with the logic required to create an ADI)
- Is there some refactoring to do? (hint: yes but that is out of scope for this workshop)
- What is the impact (scope) of the refactoring?

This lab will be using an IBM zTrial instance of Application Discovery and Delivery Intelligence (ADDI) running on an IBM cloud.

This ADDI zTrial instance will be running within an IBM Developer for z/OS® (IDz)

You will be provided with a unique URL and Log in information to access this zTrial instance

No prior IDz or COCOL programing experience is required for this Lab.

Start your zTrial instance.

In this step, **you will access your personal ADDI instance in IBM's Cloud**

1 Please use the unique URL and PASSWORD assigned to you

2 Your id is Administrator

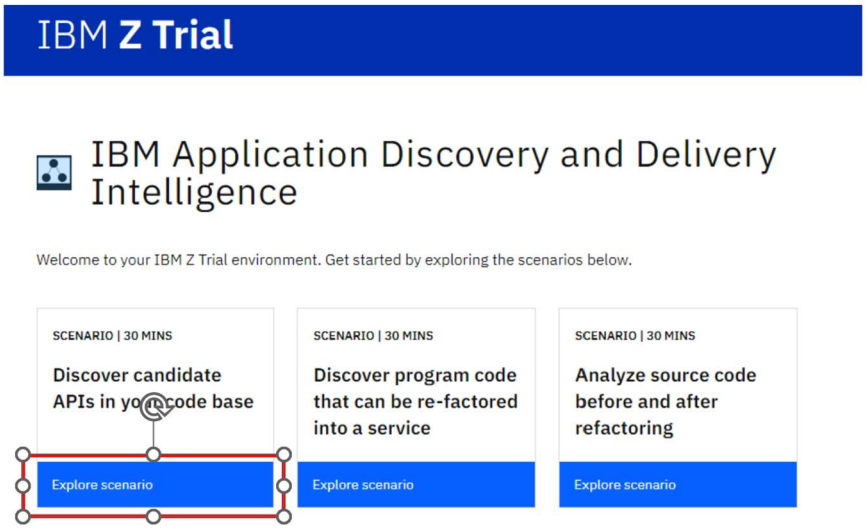
Web browser sign-in details EXAMPLE (Do NOT use this URL)

URL: https://T-8592-169-59-184-212.ibmztrialmachines.com/
User name: Administrator
Password: ih70pxPEDOm7vUcT22ya

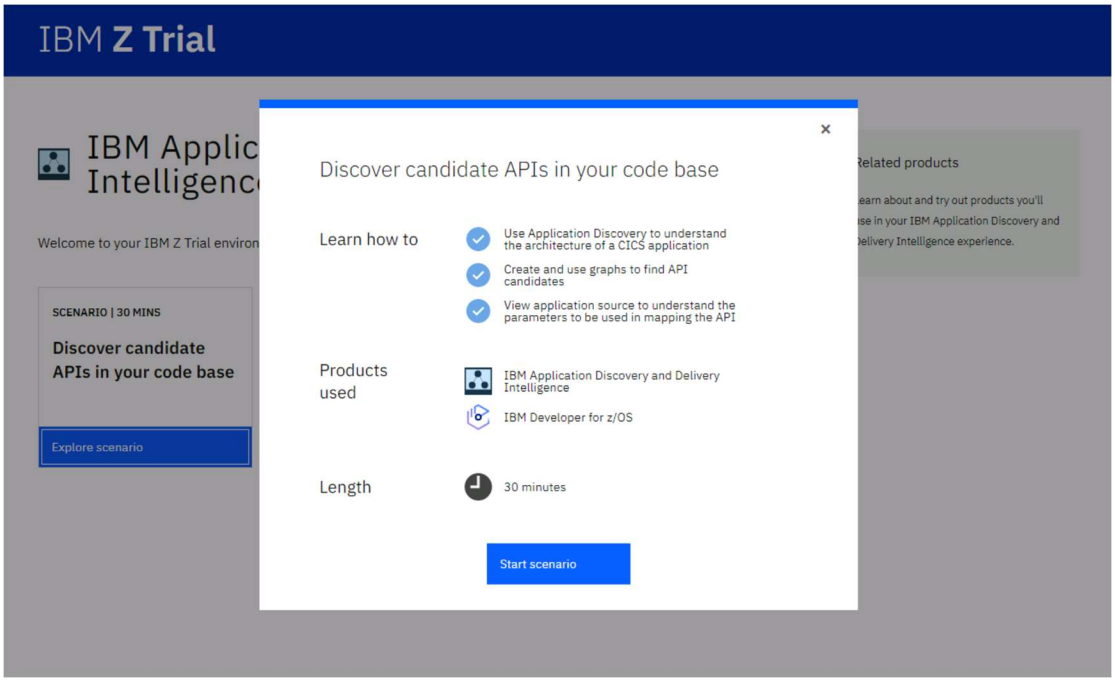
We recommend viewing your trial in full screen mode.
If you have any problems connecting to your trial, please email us at ztrial@uk.ibm.com for assistance.

3 You will be logged into the ADDI zTrial instance

Click the blue Explore scenario box under the "Discover candidate API's in your code base

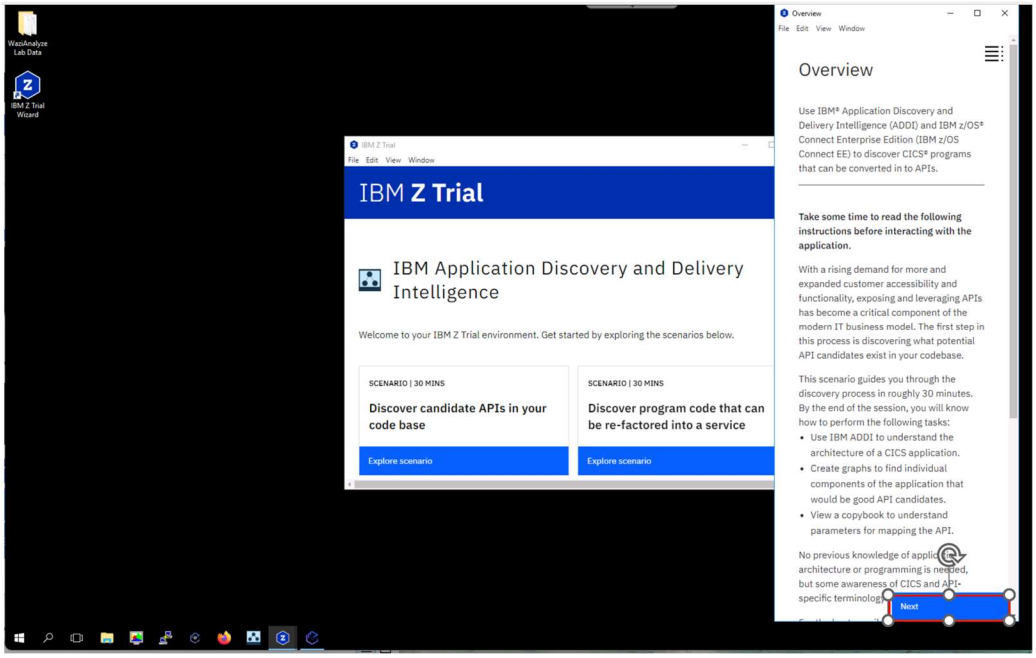


4 Click the blue Start scenario box under the "Discover candidate API's in your code base

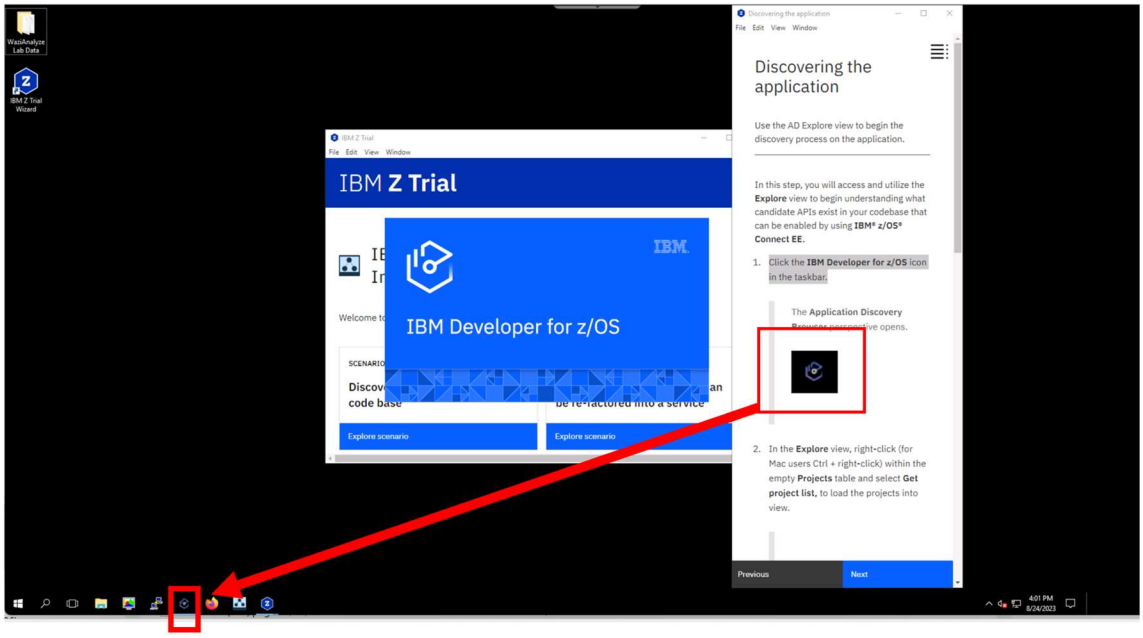


5 The ONLINE version of scenario instructions will appear.

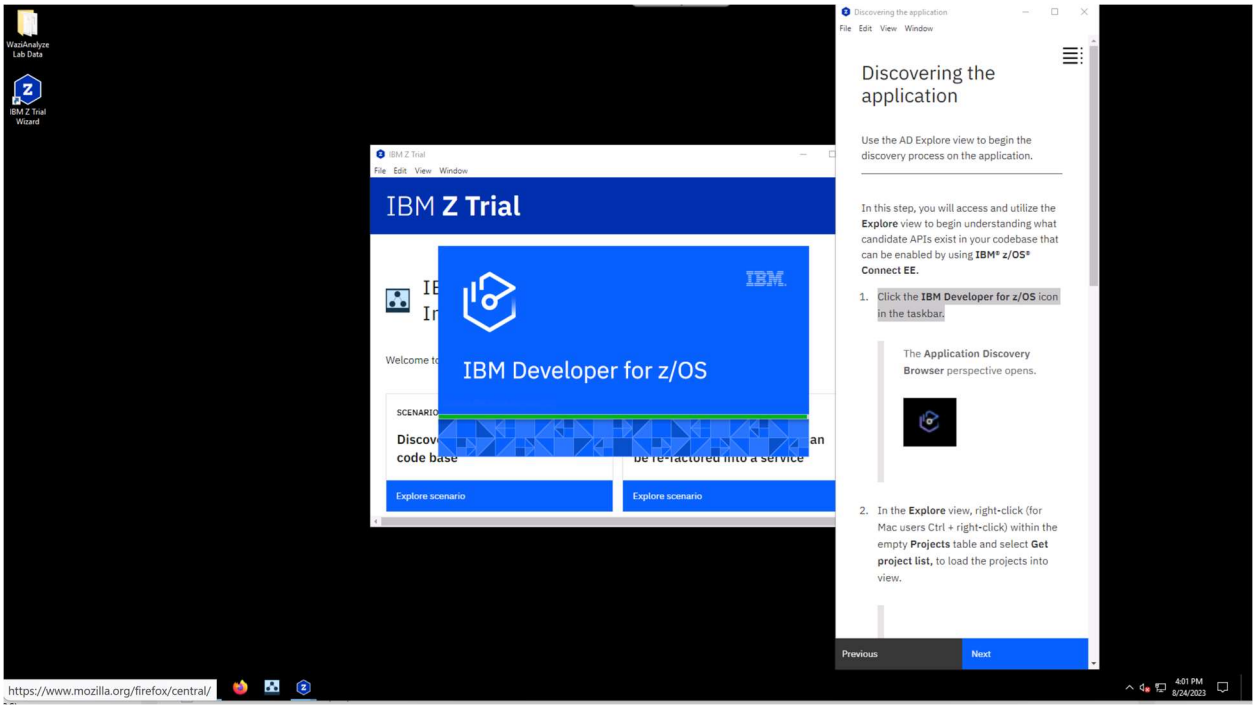
Click the blue Next in the Online scenario



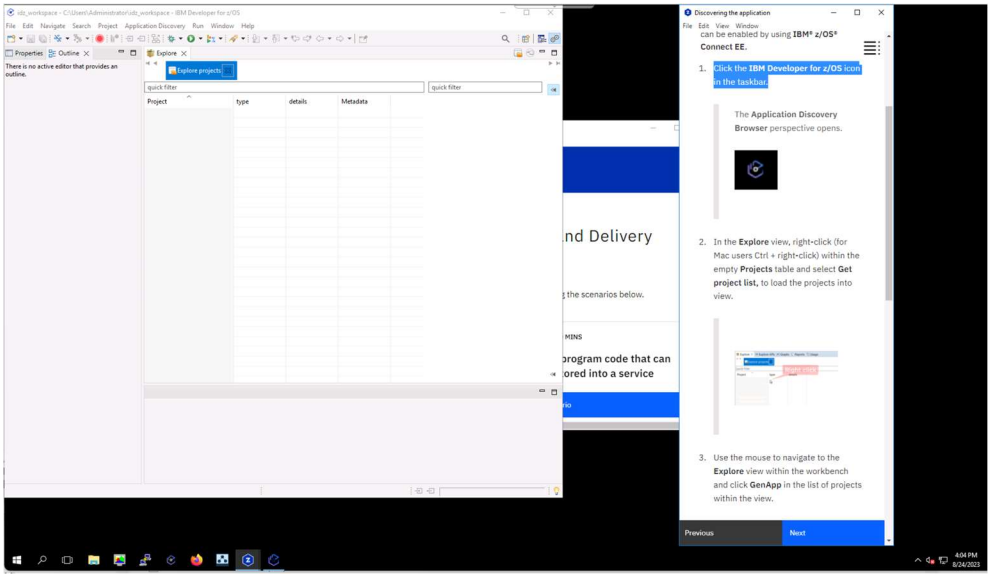
6 Click on the Application Discovery icon on you task bar 



7 The ADDI Analyze client perspective will start up



PLEASE wait a minute until IDz starts with the ADDI Analyze Browser perspective and shows the following Project screen.



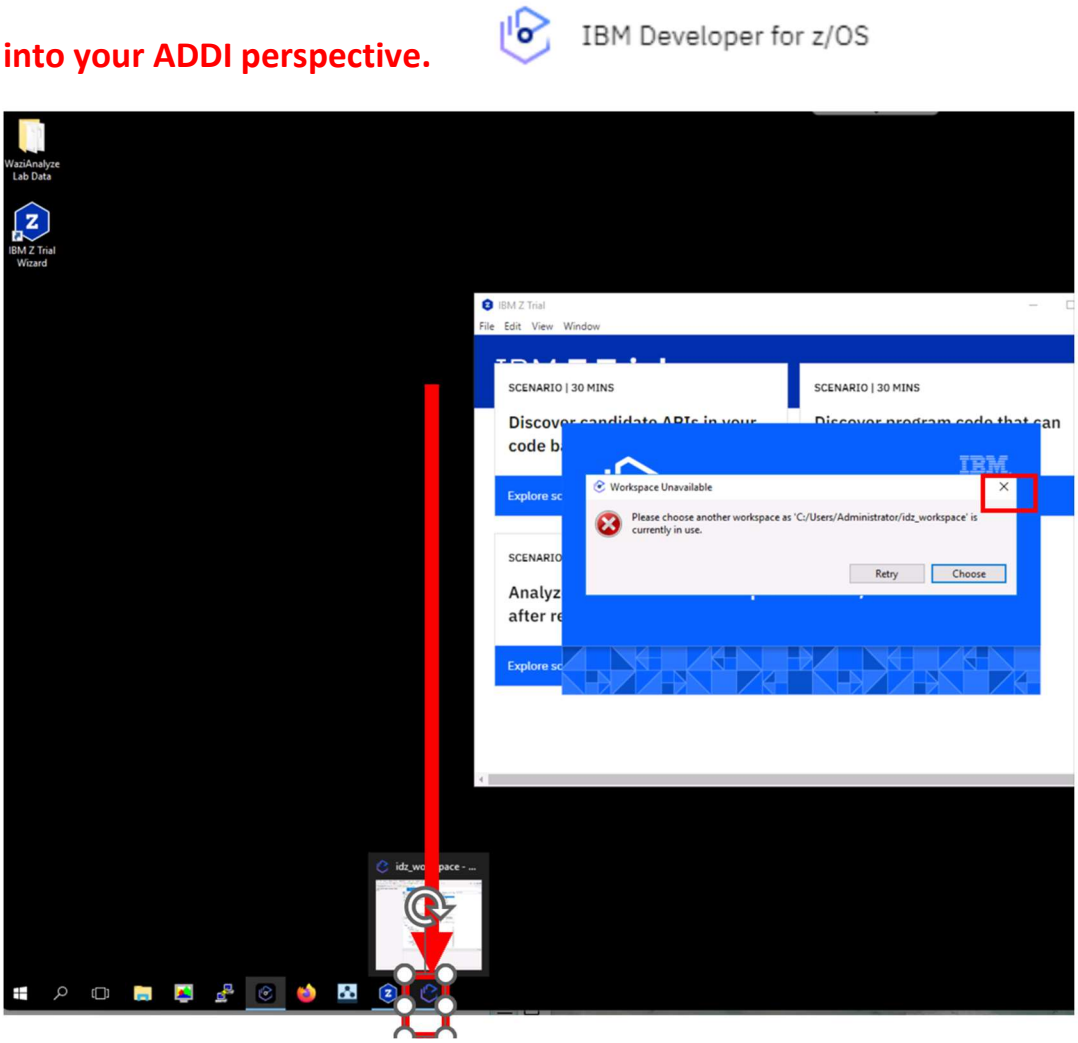
NOTE 1: From this point on, these printed workshop scenario instructions differ from the online version. Please use these printed instructions. You can request a new zTrial instance of your own after this session and try the other online scenarios or try the online version later if you like.

NOTE 2: If the following error pops up anytime from this point on during your workshop, it means that your IDz/ADDI is already running.

Do the following to get back into ADDI

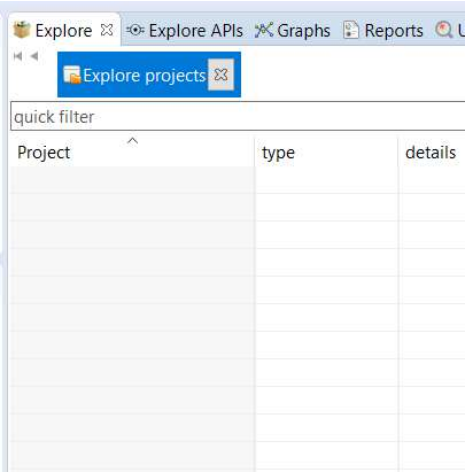
-Close down the error popup

- Click the running IDz workspace icon on your task bar to get back into your ADDI perspective.

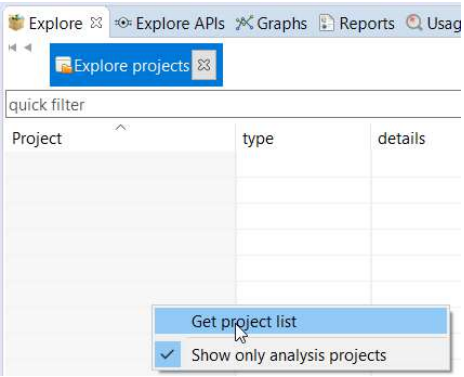


Start the ADDI Analyze client and see the projects that have been built.

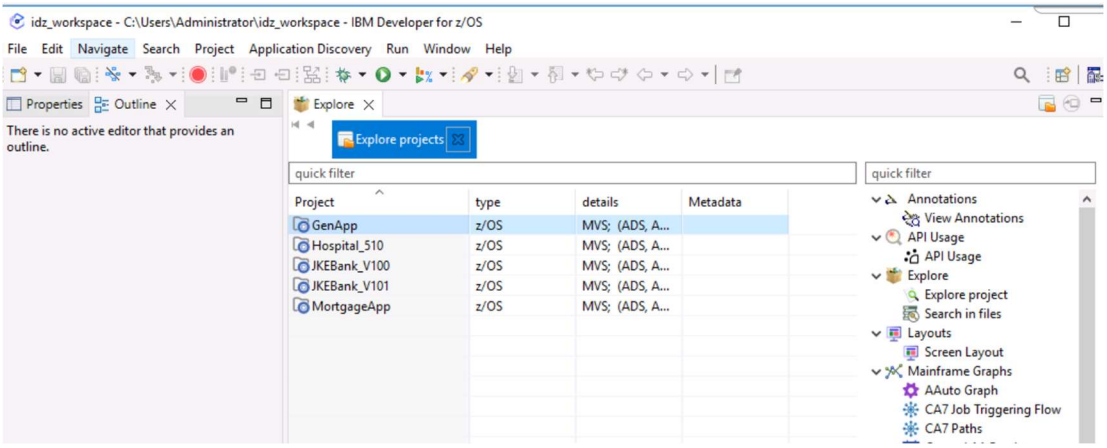
In this step, you will access and utilize the Explore view to begin understanding what candidate APIs exist in your codebase that can be enabled by using IBM® z/OS® Connect EE.



To populate the list of projects, right click anywhere in the list and click on **“Get project list”.**



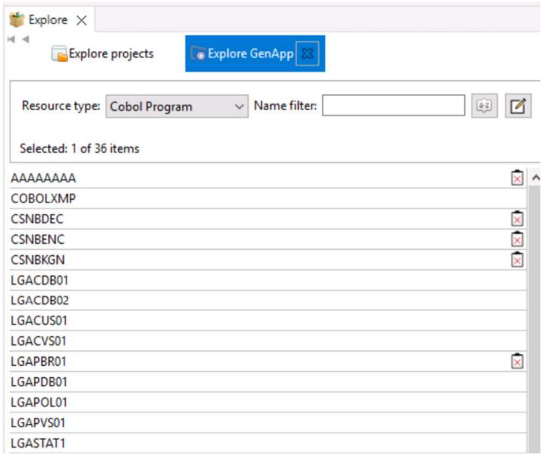
The browser will now contain a list of projects to explore. We will focus on two: **MortgageApp** and **GenApp**. **Note** that there may be other applications/projects shown.



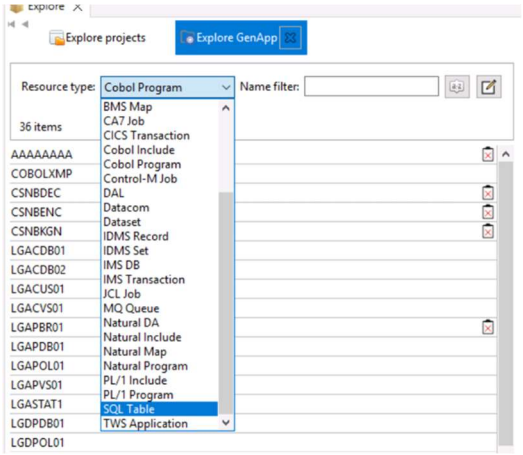
Identify how the customer table is used.

We want to find the usages of the Customer table and find the code that reads that table so we can understand the logic and expose it as a service.

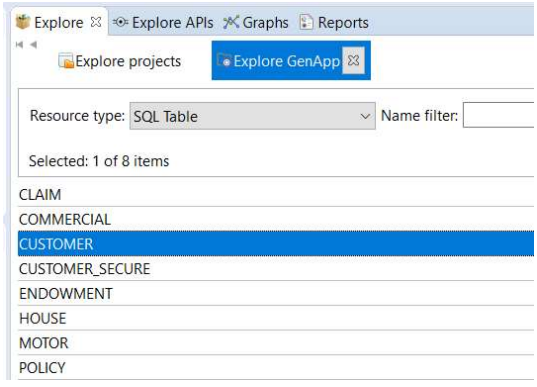
Double click on **GenApp**, this will open another pane.



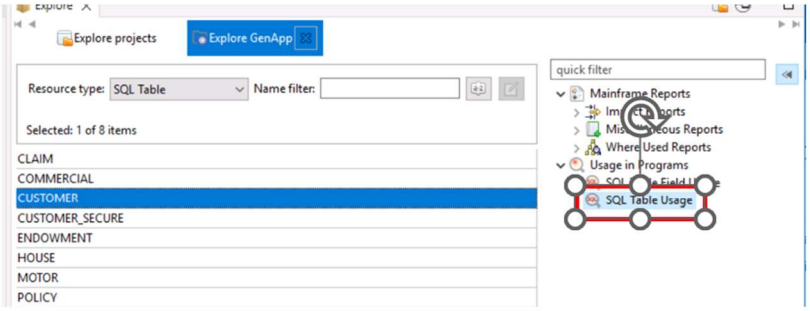
In **Resource Type drop down**, select **SQL Table.**



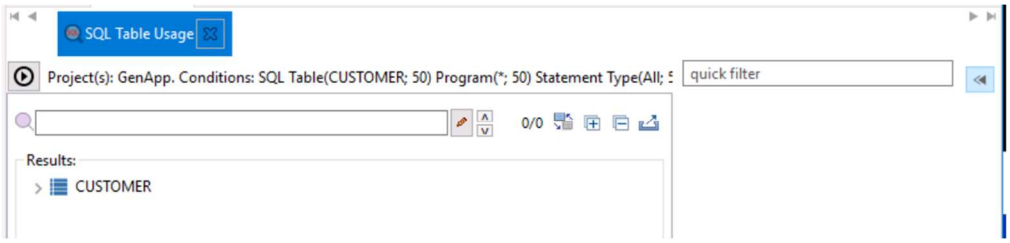
A list of tables is displayed. Select **Customer.**



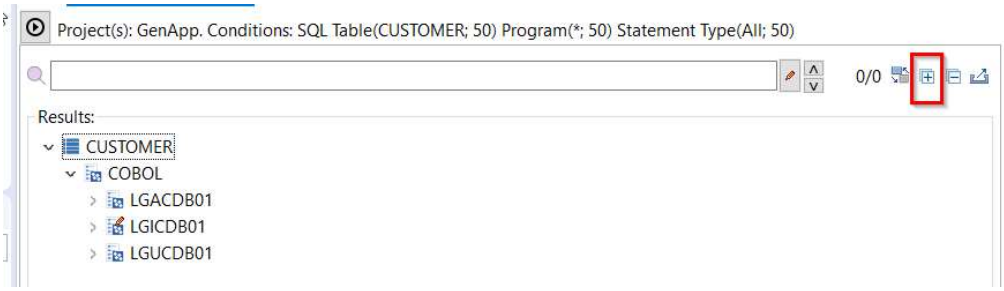
On the right, you see a list of possible actions that you can perform on the resource. Select (Double click) **SQL Table Usage**.



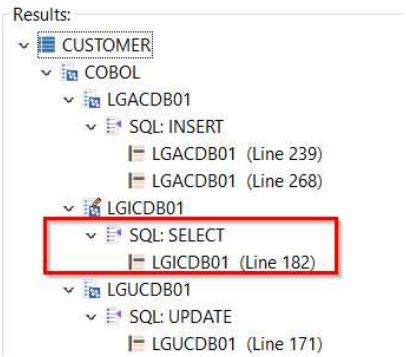
This opens the **SQL Table Usage** filtered on the **Customer table.** Note: You can also use the same view, non-filtered, from the project. This is a different workflow to reach the same information.



Expand by pushing the **+ icon (Expand all)** in the editor.

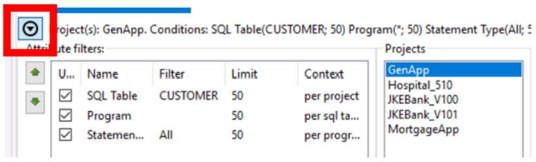


We see different levels of nodes. One of them is the **statement type**

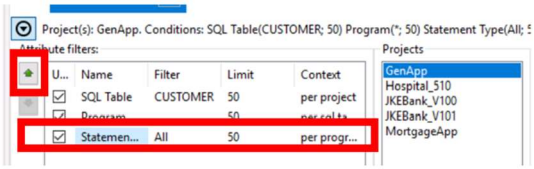


Look for the programs that are doing a **select** in the customer table. There is a single entry. In this view, you can change the order of nodes in the settings.

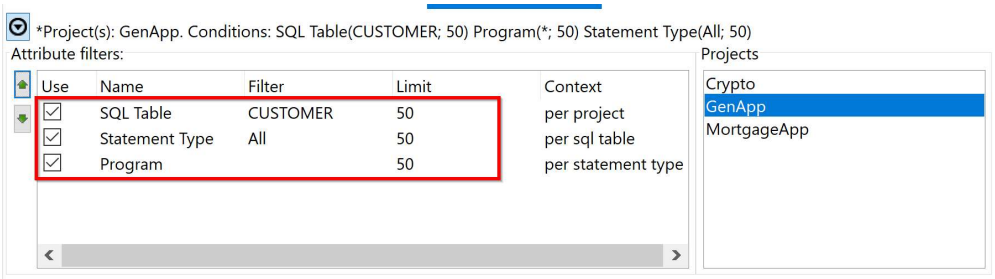
Click the arrow to expand the settings.



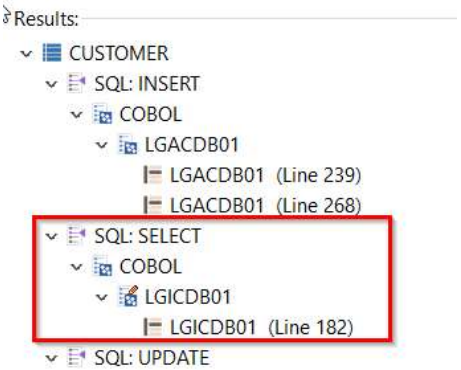
Select **Statement Type** and move it up 1 level with the **up arrow**:



Notice the order of criteria has changed (**Statement Type** is listed before **Program**).

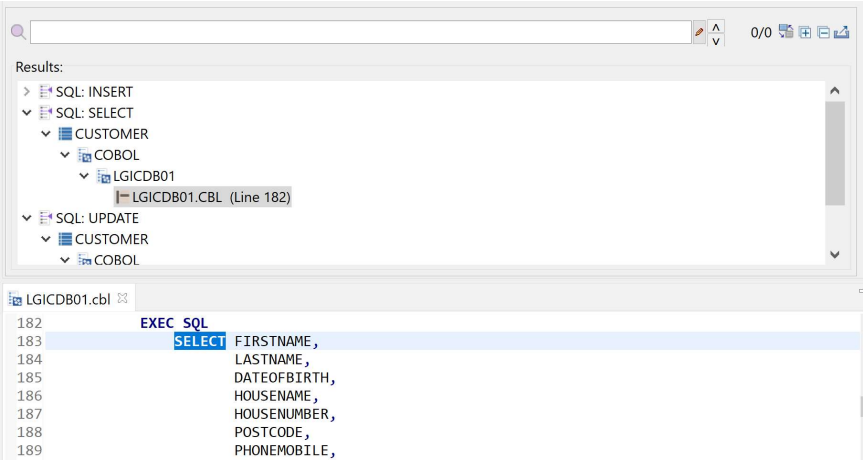


Expand the nodes



The order of nodes has changed. The nodes are now grouped by Table, then Programs by Access Type performing that statement.

Double click on the **entry with (Line 182)**. An editor opens and highlights the source code statements. It is a **SELECT** statement.



NOTE: We found that Program LGICBD01 is the only program that is doing a SQL SELECT against table CUSTOMER. We can now try to determine where (or how) this program gets the data to know what data is required to return the correct data

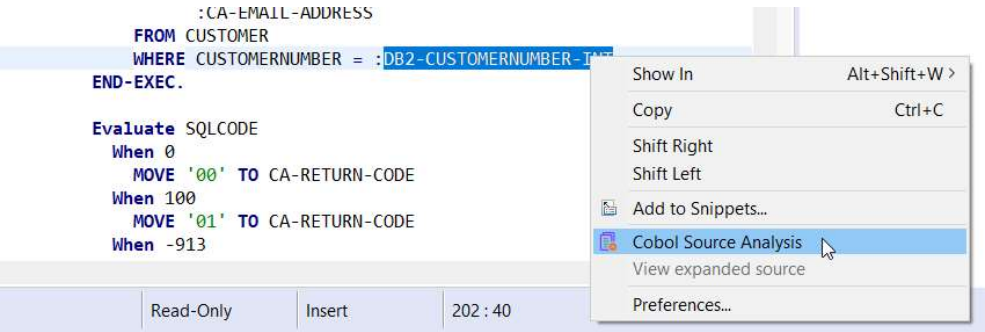
Understand how the parameter of the select is obtained

We want to understand how the parameters of this query are obtained in this program. Scroll to the bottom of the highlighted text to look at the WHERE clause of the SQL statement:

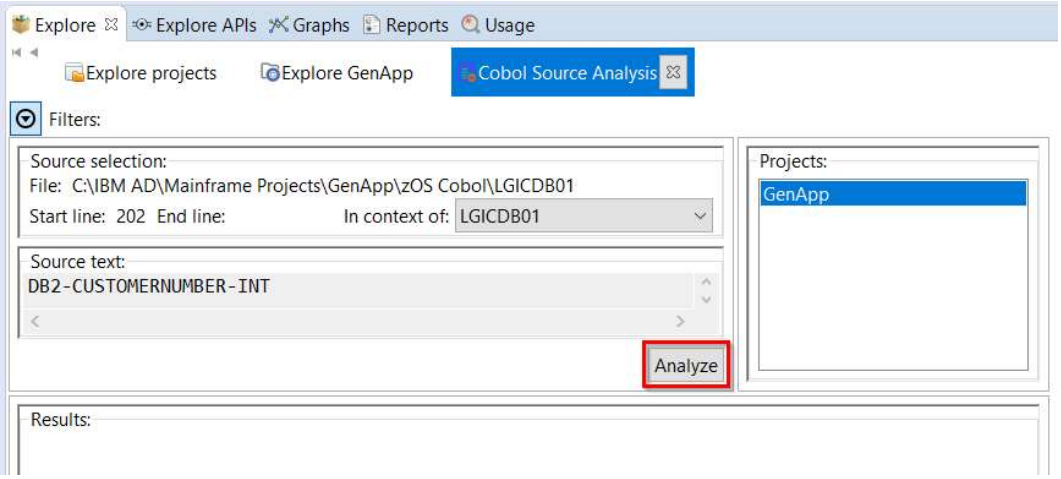
```
197                     :CA-POSTCODE,
198                     :CA-PHONE-MOBILE,
199                     :CA-PHONE-HOME,
200                     :CA-EMAIL-ADDRESS
201                FROM CUSTOMER
202                WHERE CUSTOMERNUMBER = :DB2-CUSTOMERNUMBER-INT
203             END-EXEC.
```

Next, we want to trace the host variable used here: DB2-CUSTOMERNUMBER-INT.

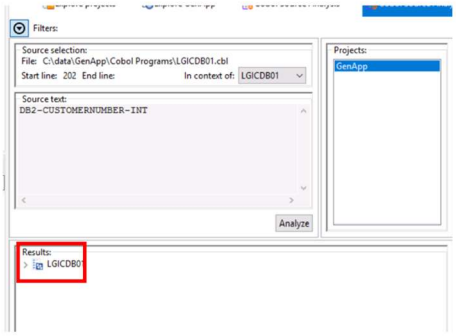
Highlight the host variable name (DB2-CUSTOMERNUMBER-INT) and **right click** on it. We will perform a COBOL Source Analysis.

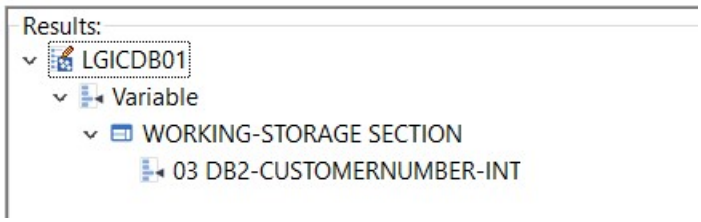


This panel opens. **Press** the ***Analyze*** button.



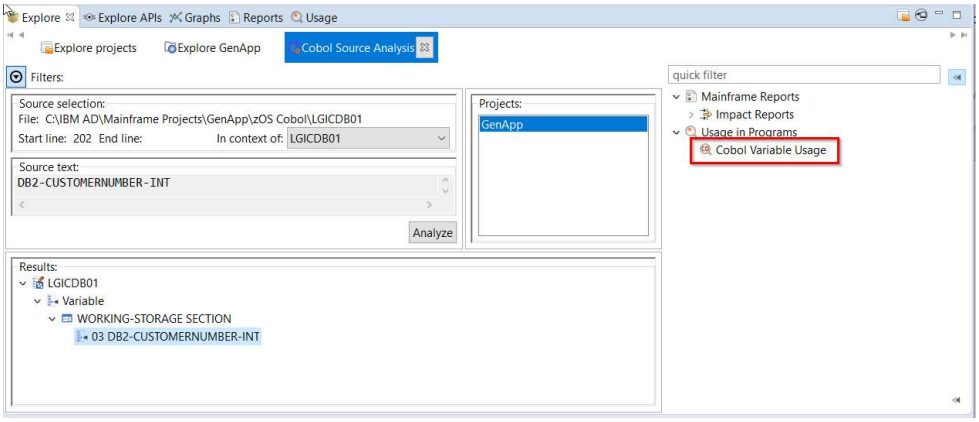
Expand the results:



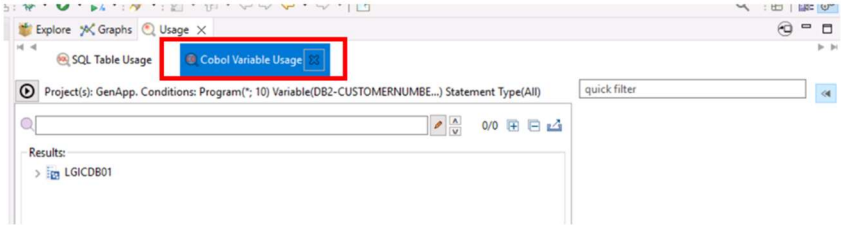


The host variable is declared in **the working storage** section of the program. Now let's try to find how this variable is used.

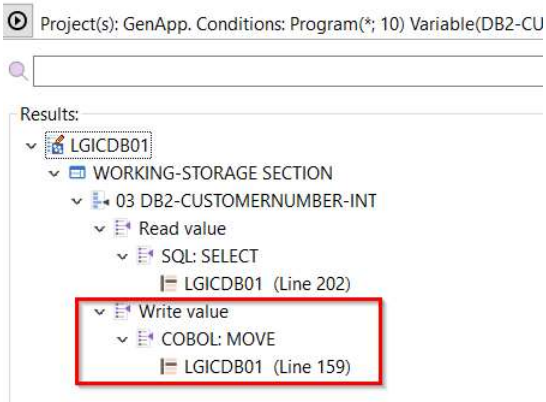
Click on the host variable (DB2-CUSTOMERNUMBER-INT) in the Results. Options will appear in the upper right panel. You will see the actions you can perform on the selection. Let's select and perform a **Cobol Variable Usage.** Double click on Cobol Variable Usage to open.



This view opens.

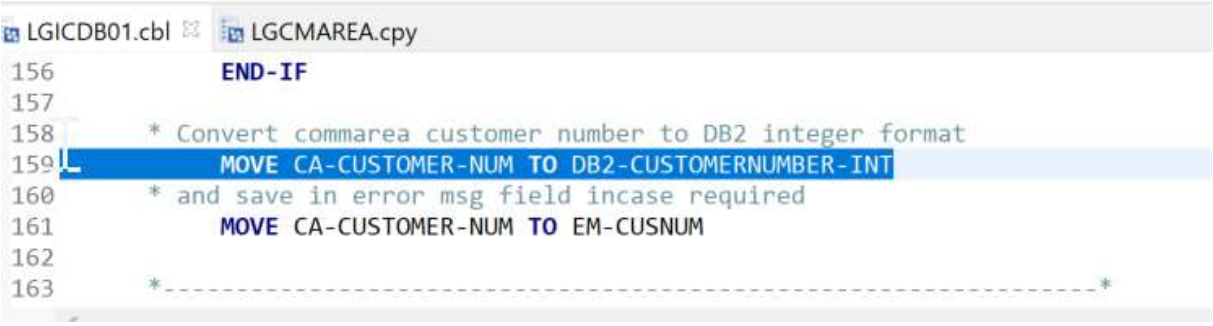


Expand the **nodes.**



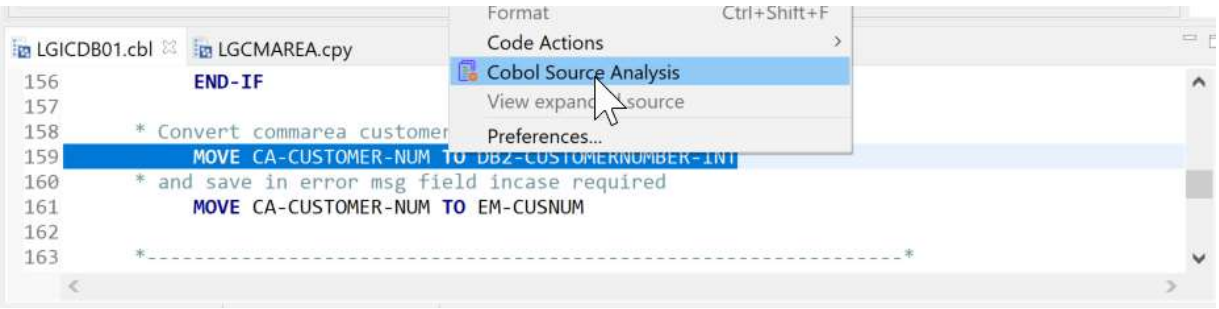
The **variable** is being written through a ***'Move'*** statement.

Let's look at it by double clicking the (Line 159) line:

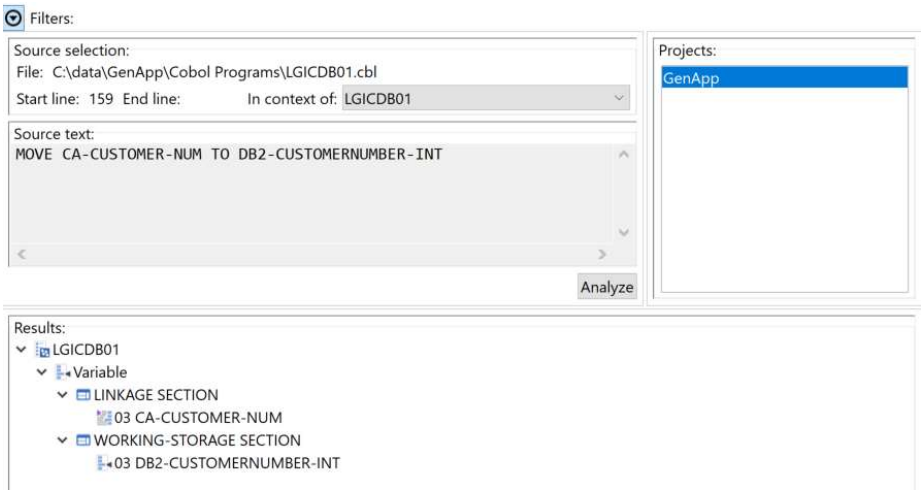


We need to find out more about **CA-CUSTOMER-NUM** now.

Highlight the **entire line** and **right click** on it. Select from the menu to perform another **Cobol Source Analysis**.



Click Analyze. Then expand the Results, you will see that the value of the field comes from the linkage section.



This means that the data is provided by another program. To find the other program, we want to find which program is calling the program **LGICDB01.**

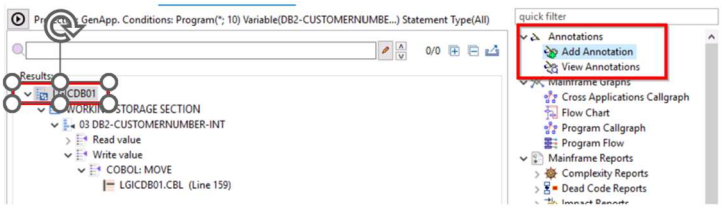
In summary, we have determined.

1. **LGICDB01** is the program that reads the Customer table
2. Host variable's value is passed from another program. We need to determine what program that is.

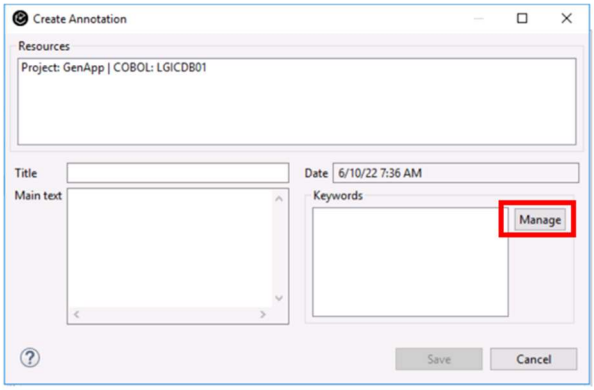
Let's pause and capture what we found. To do this, we use the **annotate** feature to document.

Annotate the program for future reference

Select the **LGICDB01** program and then **choose** (double click) ***"add an annotation"***.

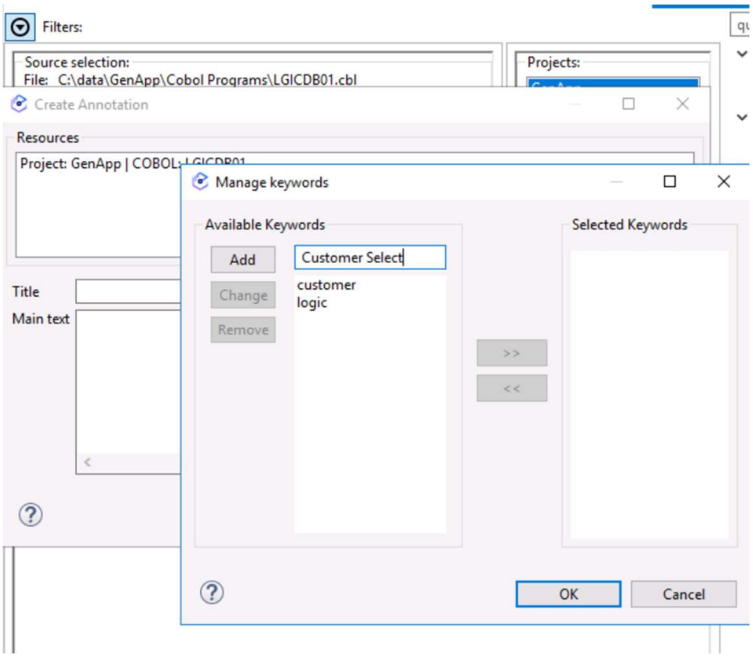


The **Create Annotation** panel opens.

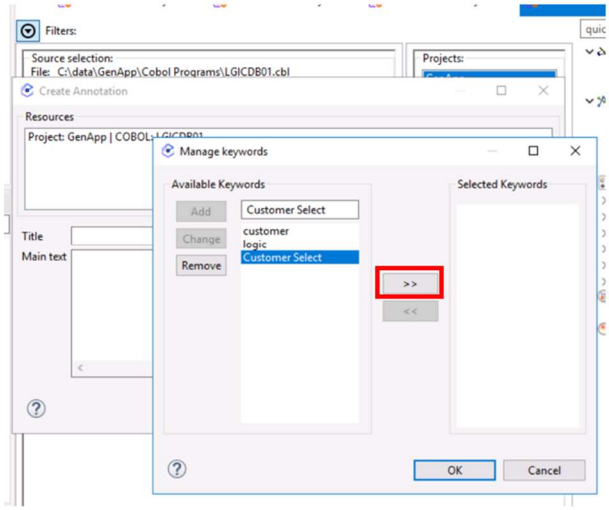


You will see that someone has already created some annotations. We would typically investigate what these are to see what other information was captured and if we need to create a new one or update the existing but for now, we will create a new one for our analysis. Let's create a keyword (to be used later).

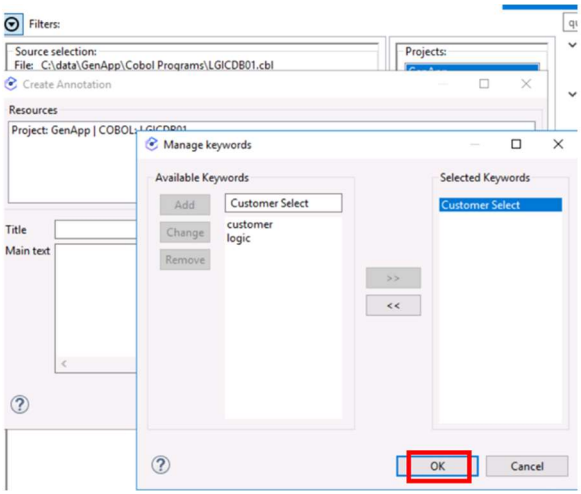
- Click **Manage**
- **Type** in "***CUSTOMER Select***"



To apply our keyword (CUSTOMER) to our program, **Select *Customer Select*** and move it to the Selected Keywords

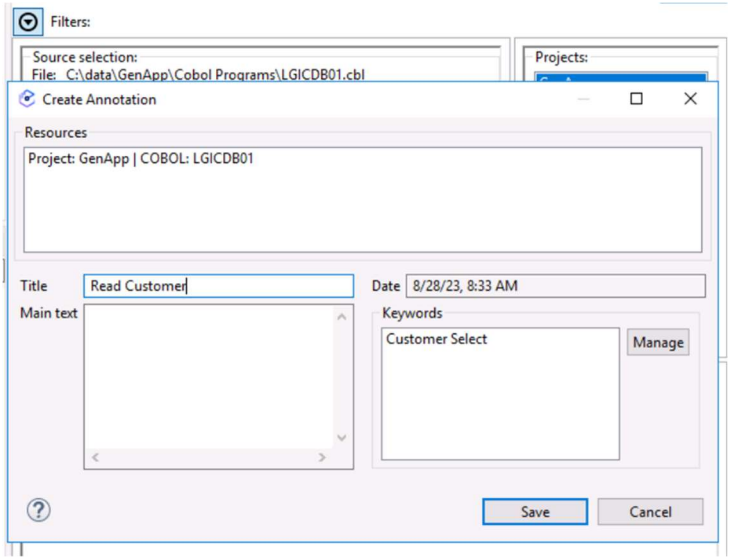


Press the ***OK*** button.



Next, we need to give this Annotation a Title so we can save it.

Type *Read Customer* and click the ***Save*** button.



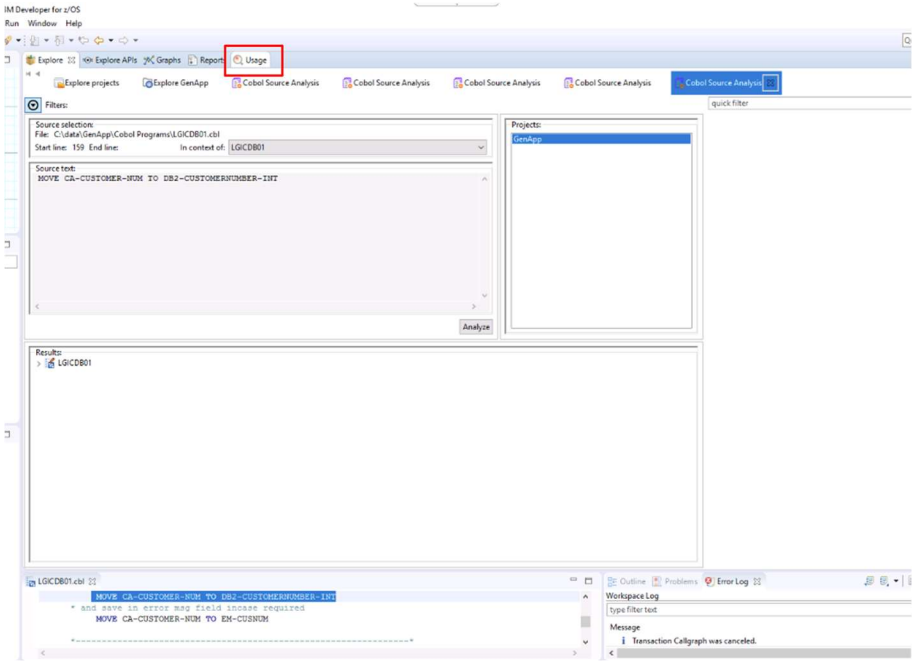
NOTE: This is optional, but it might be useful to enter information in the **Main text** box as to why this annotation was made.

Congratulations! We have created and saved the annotation. This can be used to capture and share information that you have learned from your analysis so that others might not have to redo all of your analysis.

Impact analysis report

We' have done a lot of work and have several results tabs open. If you have not closed any we can navigate back to them in order to review, export information or perform additional analysis. We want to go back to the SQL table usage view. 1st we will click the Usage Tab to get back to all of our Usage

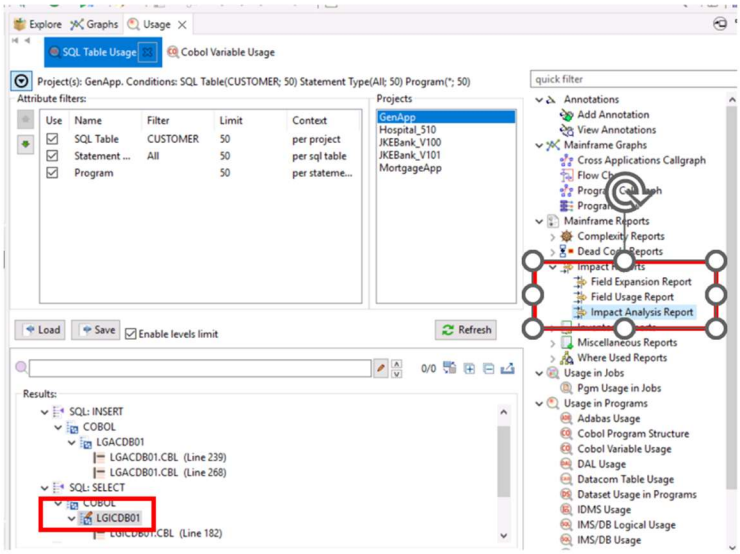
Result Views. Click the **Usage** tab.



Make sure you are in the **SQL Table Usage** tab

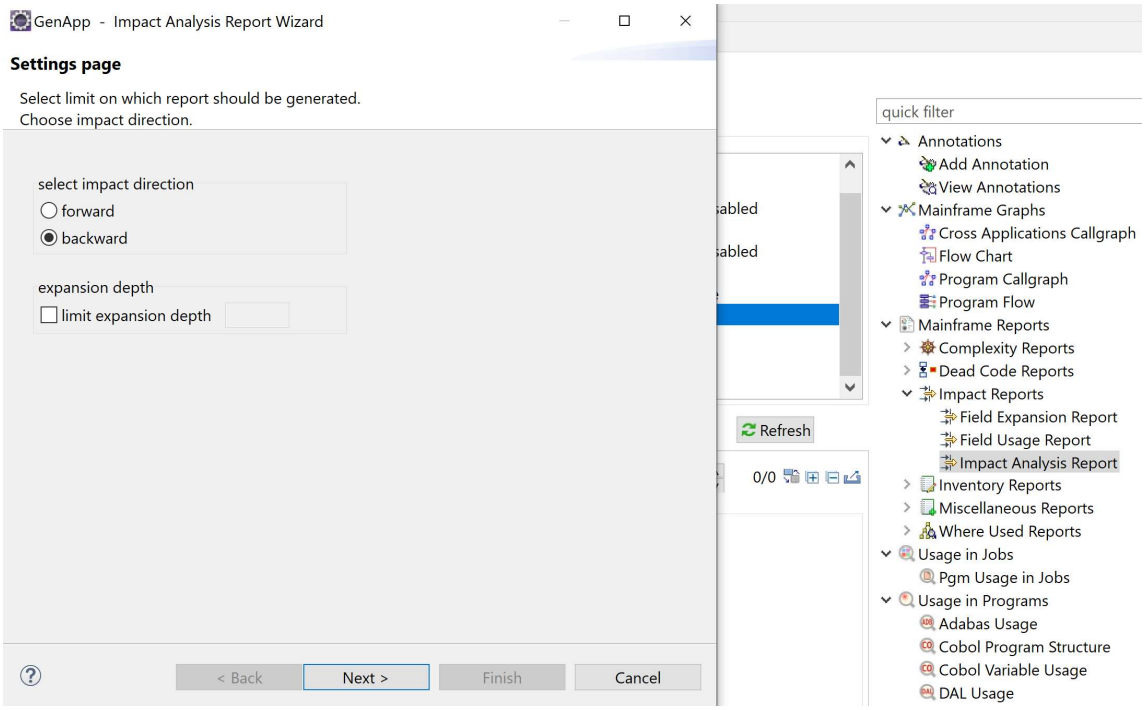
> ***Note:*** *If you have closed or can't find the SQL Table Usage tab, you can recreate it by redoing the* ***"Identify how customer table is used"*** *step you did earlier in this workshop.*

Select the program (LGICD801) and select Impact Analysis Report on the right panel (under Impact Reports):



Let’s see how the data gets passed to this program.

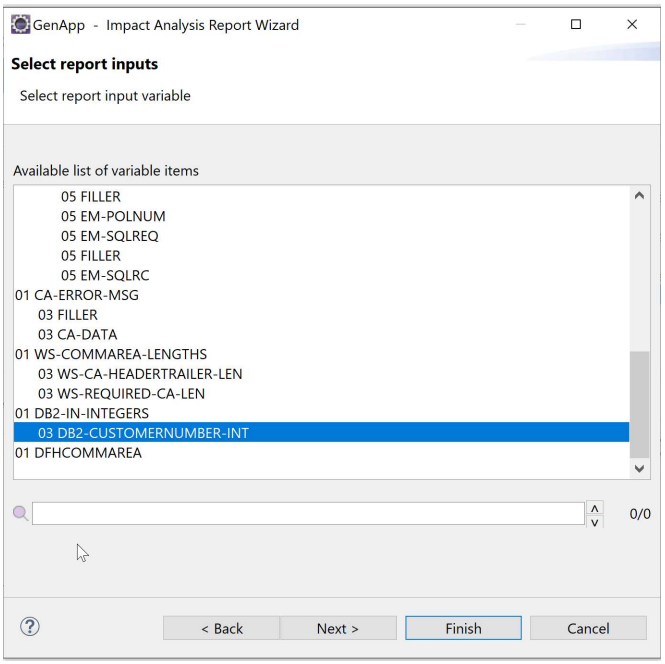
Select the direction of the impact to be performed (here we will select **backward**). Click **Next**.



You will choose the host variable of the select statement.

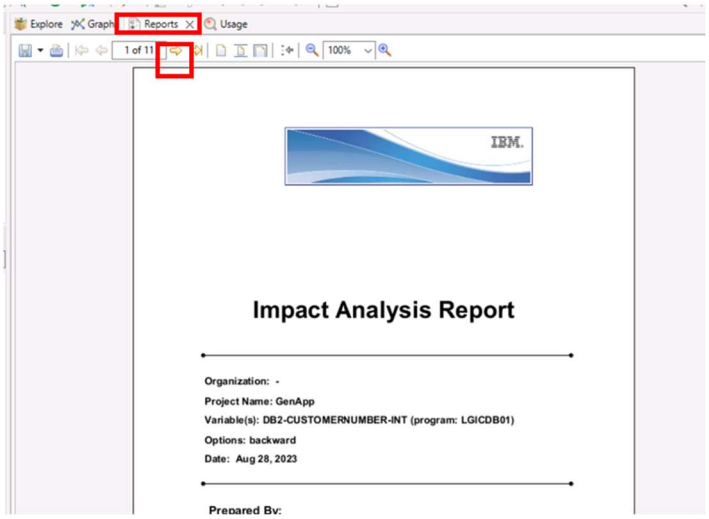
Scroll to the bottom of the list or start typing DB2 in the search to get to DB2-CUSTOMERNEMBER-INT variable.

Click **Finish**. We want to understand how it is passed to the SQL statement:



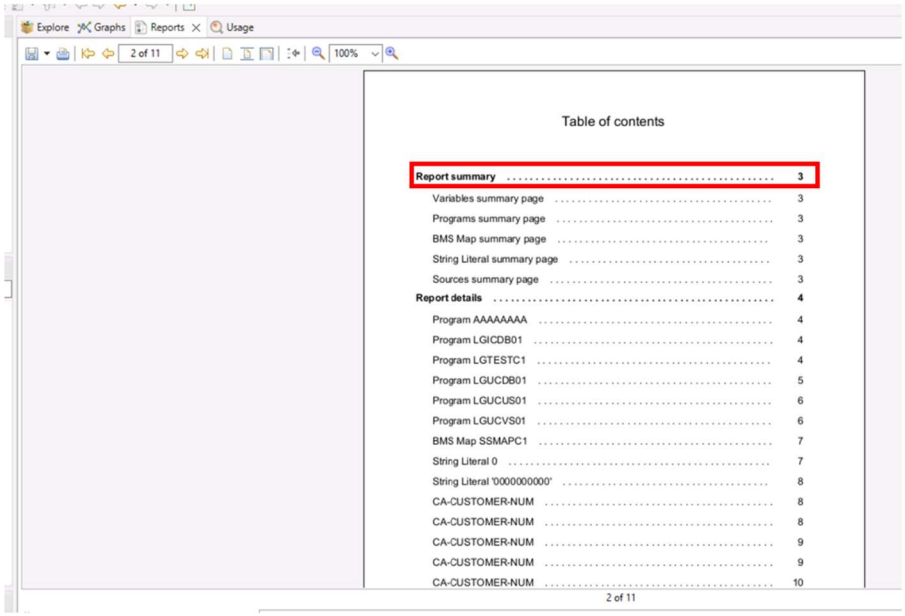
You are now in the Reports Tab where you can explore the report.

You can use the Yellow arrows or page up/down on your keyboard or hyperlinks to change pages



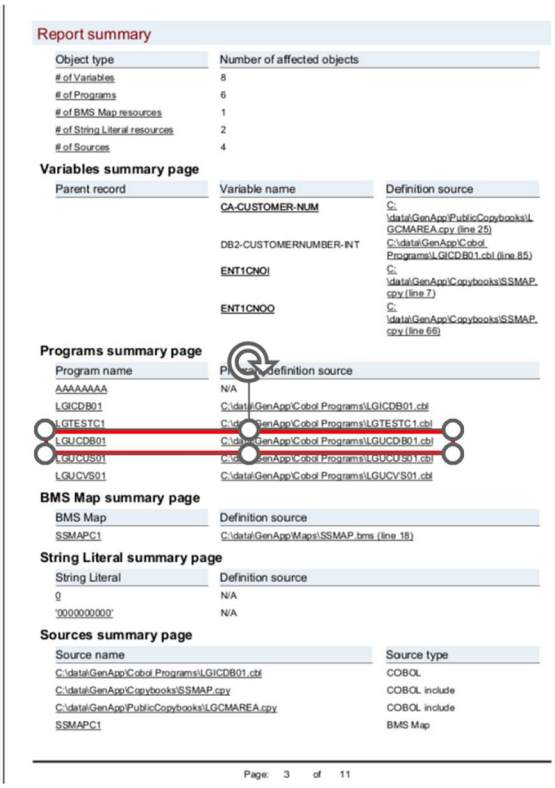
Advance to the Table of contents page.

Notice the types of data available listed in the Table of Contents of this report. The data is hyperlinked to other data within this report. Click on the **Report summary** by clicking on the hyberlink to advance to the summary page.



See that LGTESTC1 is involved in the data flow of the data Element.

Lets' drill down by clicking on the program



Report summary

Object type	Number of affected objects
# of Variables	8
# of Programs	6
# of BMS Map resources	1
# of String Literal resources	2
# of Sources	4

Variables summary page

Parent record	Variable name	Definition source
	CA-CUSTOMER-NUM	C:\data\GenApp\PublicCopybooks\LGCMAREA.cpy (line 25)
	DB2-CUSTOMERNUMBER-INT	C:\data\GenApp\Cobol Programs\LGICDB01.cbl (line 85)
	ENT1CNOI	C:\data\GenApp\Copybooks\SSMAP.cpy (line 7)
	ENT1CNOO	C:\data\GenApp\Copybooks\SSMAP.cpy (line 66)

Programs summary page

Program name	Program definition source
AAAAAAA	N/A
LGICDB01	C:\data\GenApp\Cobol Programs\LGICDB01.cbl
LGTESTC1	C:\data\GenApp\Cobol Programs\LGTESTC1.cbl
LGUCDB01	C:\data\GenApp\Cobol Programs\LGUCDB01.cbl
LGUCUS01	C:\data\GenApp\Cobol Programs\LGUCUS01.cbl
LGUCVS01	C:\data\GenApp\Cobol Programs\LGUCVS01.cbl

BMS Map summary page

BMS Map	Definition source
SSMAPC1	C:\data\GenApp\Maps\SSMAP.bms (line 18)

String Literal summary page

String Literal	Definition source
0	N/A
'0000000000'	N/A

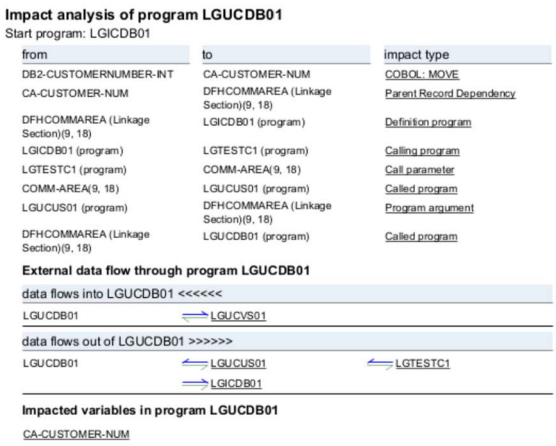
Sources summary page

Source name	Source type
C:\data\GenApp\Cobol Programs\LGICDB01.cbl	COBOL
C:\data\GenApp\Copybooks\SSMAP.cpy	COBOL include
C:\data\GenApp\PublicCopybooks\LGCMAREA.cpy	COBOL include
SSMAPC1	BMS Map

Page: 3 of 11

We will be able to see how DB2-CUSTOMERNUMBER-INT flows through LGUCDB01 as well as what programs call or are called by it.

.



Impact analysis of program LGUCDB01

Start program: LGICDB01

from	to	impact type
DB2-CUSTOMERNUMBER-INT	CA-CUSTOMER-NUM	COBOL: MOVE
CA-CUSTOMER-NUM	DFHCOMMAREA (Linkage Section)(9, 18)	Parent Record Dependency
DFHCOMMAREA (Linkage Section)(9, 18)	LGICDB01 (program)	Definition program
LGICDB01 (program)	LGTESTC1 (program)	Calling program
LGTESTC1 (program)	COMM-AREA(9, 18)	Call parameter
COMM-AREA(9, 18)	LGUCUS01 (program)	Called program
LGUCUS01 (program)	DFHCOMMAREA (Linkage Section)(9, 18)	Program argument
DFHCOMMAREA (Linkage Section)(9, 18)	LGUCDB01 (program)	Called program

External data flow through program LGUCDB01

data flows into LGUCDB01 <<<<<<		
LGUCDB01	⇒ LGUCVS01	
data flows out of LGUCDB01 >>>>>>		
LGUCDB01	⇐ LGUCUS01	⇐ LGTESTC1
	⇒ LGICDB01	

Impacted variables in program LGUCDB01

CA-CUSTOMER-NUM

See the flow to the host variable, and the “call stack” in the context of LGTESTC1:

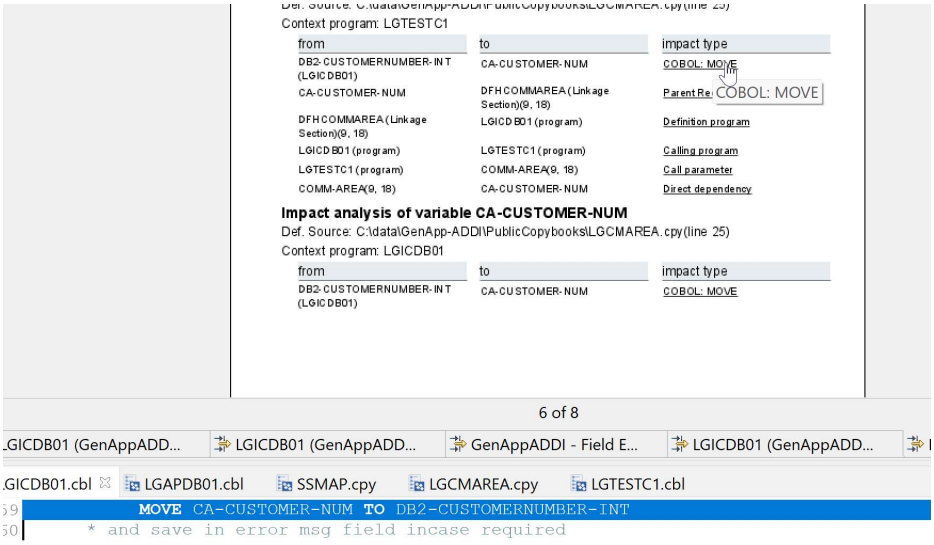
Impact analysis of variable CA-CUSTOMER-NUM

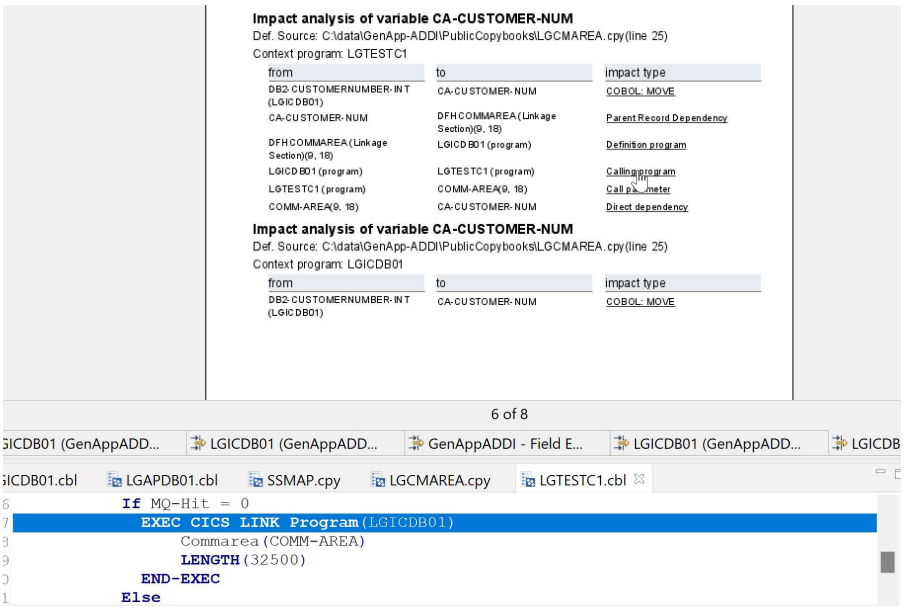
Def. Source: C:\data\GenApp-ADDI\PublicCopybooks\LGCMAREA.cpy(line 25)

Context program: LGTESTC1

from	to	impact type
DB2-CUSTOMERNUMBER-INT (LGICDB01)	CA-CUSTOMER-NUM	COBOL: MOVE
CA-CUSTOMER-NUM	DFHCOMMAREA(Linkage Section)(9, 18)	Parent Record Dependency
DFHCOMMAREA(Linkage Section)(9, 18)	LGICDB01(program)	Definition program
LGICDB01(program)	LGTESTC1(program)	Calling program
LGTESTC1(program)	COMM-AREA(9, 18)	Call parameter
COMM-AREA(9, 18)	CA-CUSTOMER-NUM	Direct dependency

Click on each line to see the statements:





What have we determined so far:

3. You now understand the flow of data and know that if you wanted to create a service or API you would need logic from both programs.
4. Host variable's value is passed from another program: LGTESTC1.
5. The flow from LGTESTC1 and that a CommArea is involved in the communication.
6. NOTE. Other online zTrial scenarios will help you identify all the COBOL logic associated with the flow of the process you were just analyzing.
7. The next portion of this workshop will show you how to visualize logic via various graphs

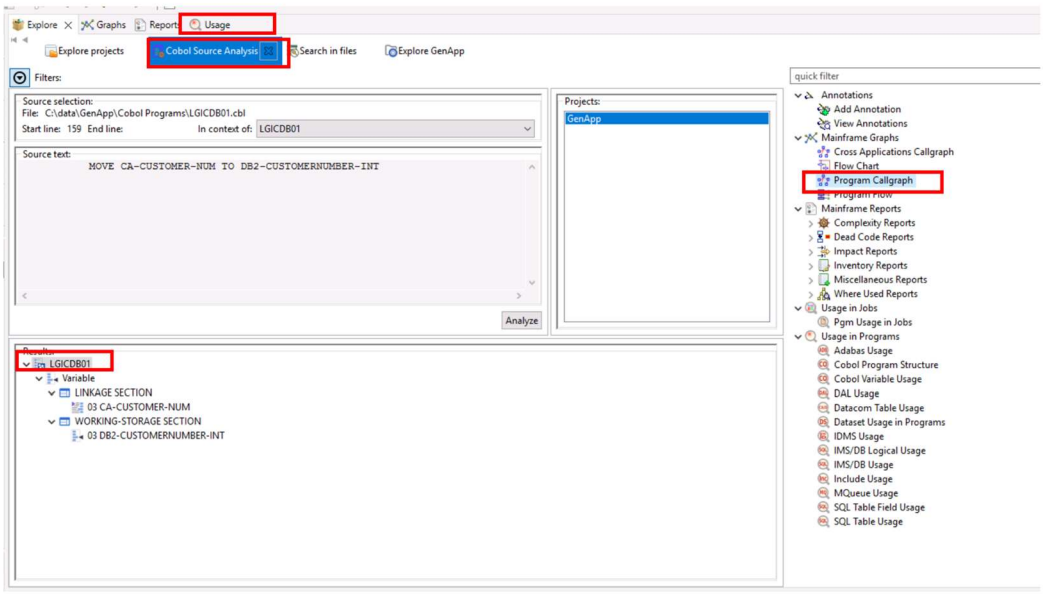
Explore Program Call Graph

Now, we want to understand which program is calling LGICDB01 with a Program Call Graph.

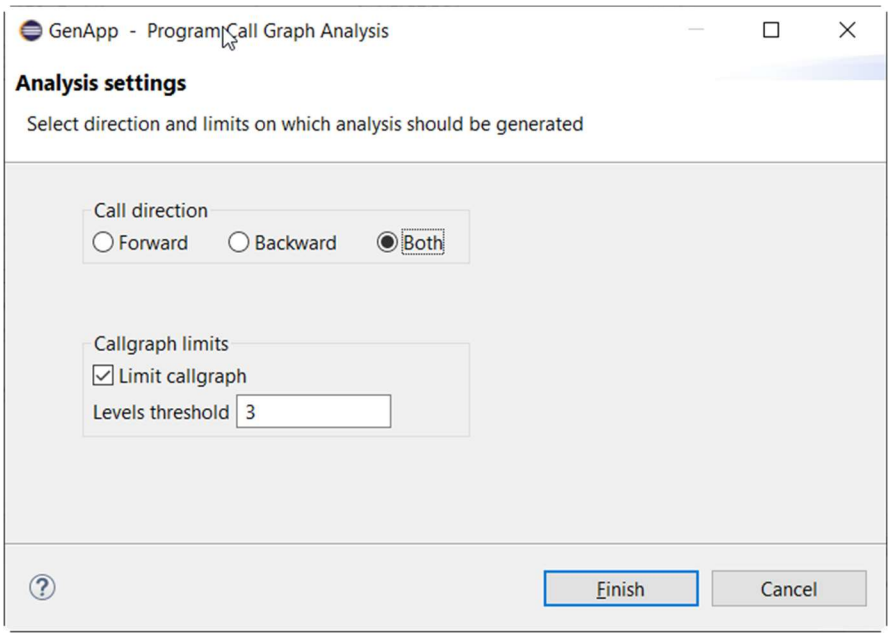
Go back to the **Usage** tab, then go to the **Cobol Variable Usage** tab.

Double click on the program.

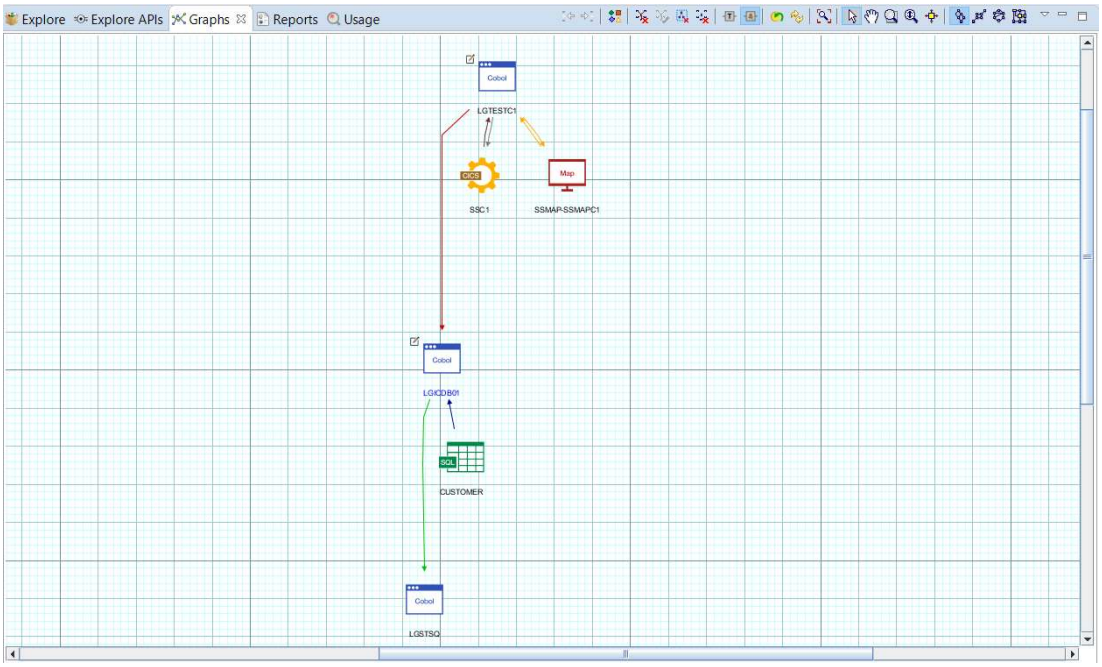
Double click on the **Program Callgraph**.



Choose ***both*** to search in both directions and **click on** the ***Finish*** button.



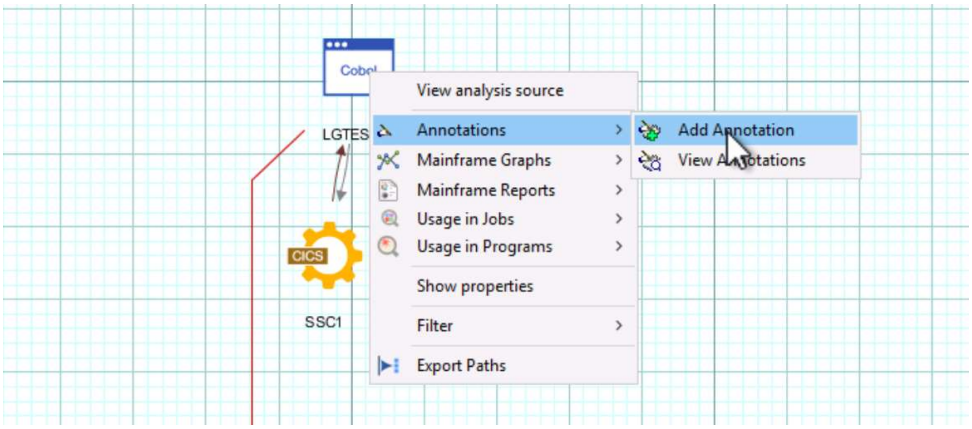
A new editor opens and displays a graph.



We see that our database program is called by **LGTESTC1** program. This program manages a map and is the entry point of a transaction.

Now that we found it, let's add an annotation.

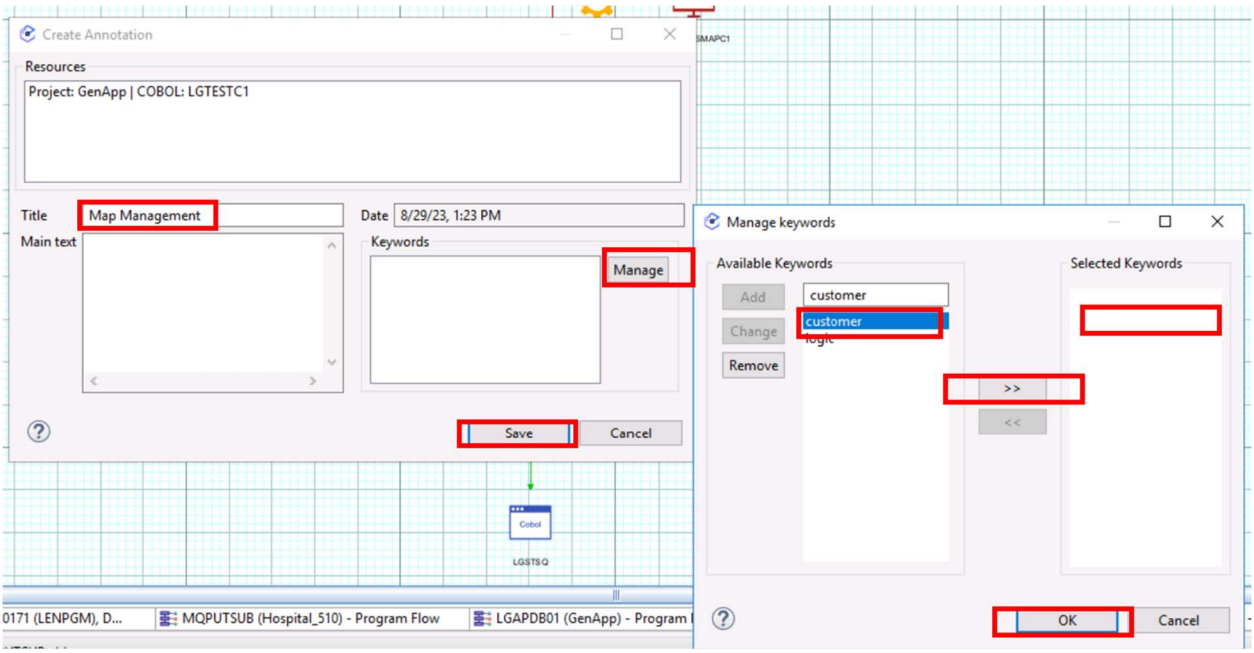
Select **LGTESTC1, right click,** and **choose "*Add Annotation*"**.



Use the same steps previously provided in this workshop to create an annotation.

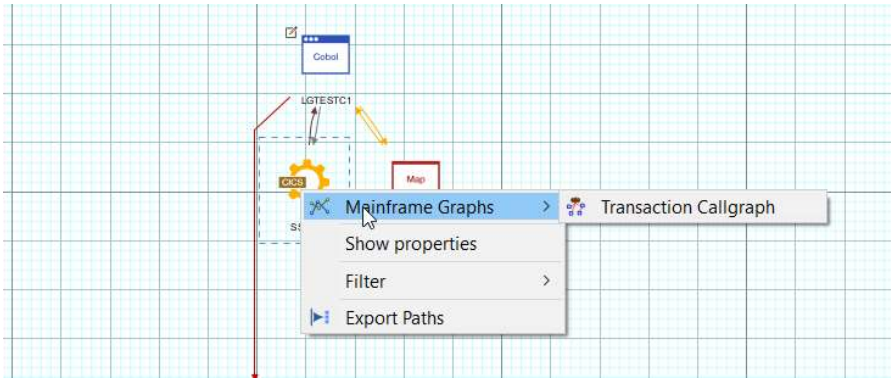
For this annotation, use **“Map Management”** for the Title.

Hint: Type Annotation’s name, click Mange button, move customer name to keyword, click ok, click save



Explore the transaction

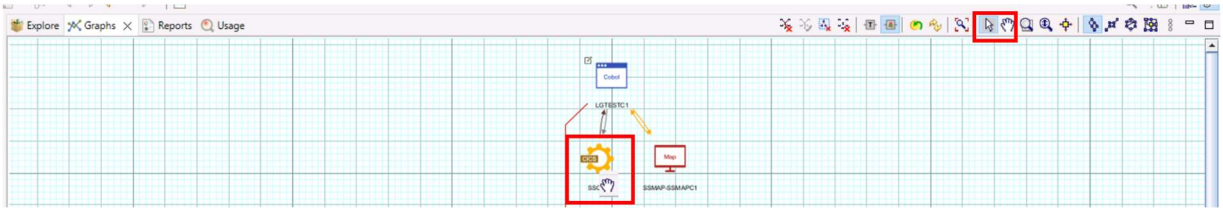
There is a transaction involved, let’s have a look. We will open a **transaction Callgraph**.



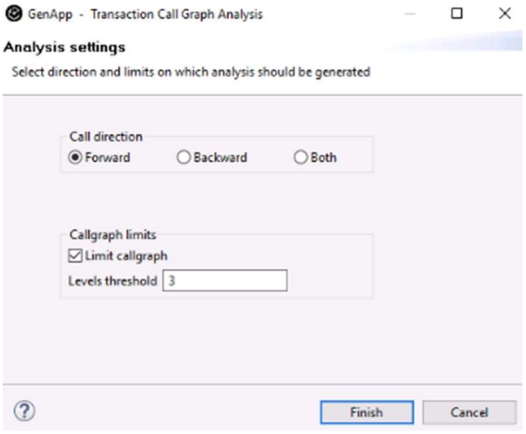
Note: If you see a  instead of a blue dotted box around the item you click on the transaction,

Your mouse is in the PAN/MOVE mode  NOT the select mode 
See how to correct this below.

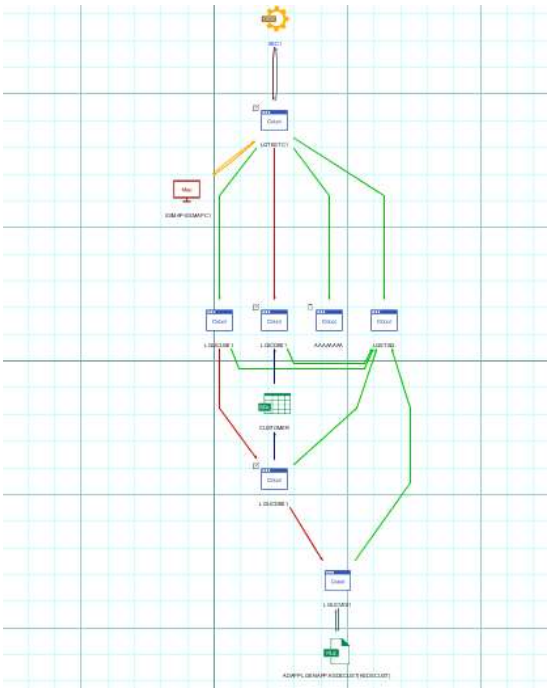
Find the toggle icons  at the top right of the graph and select the  and then click on the Transaction icon again



Choose ***both*** to search in both directions and **press** the ***Finish*** button.



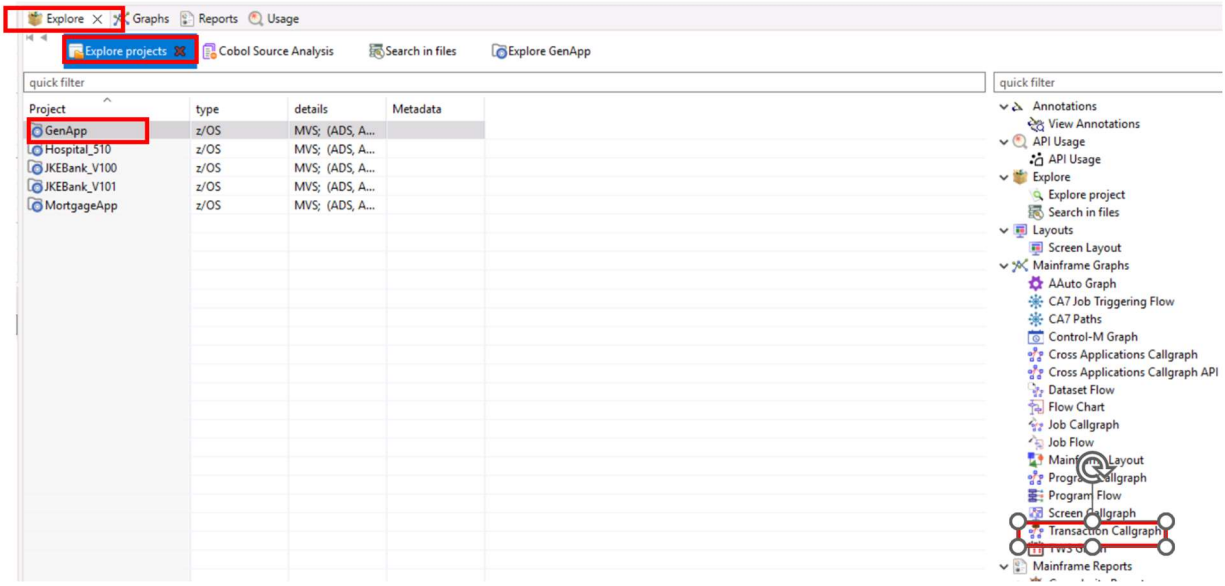
A new graph will open.



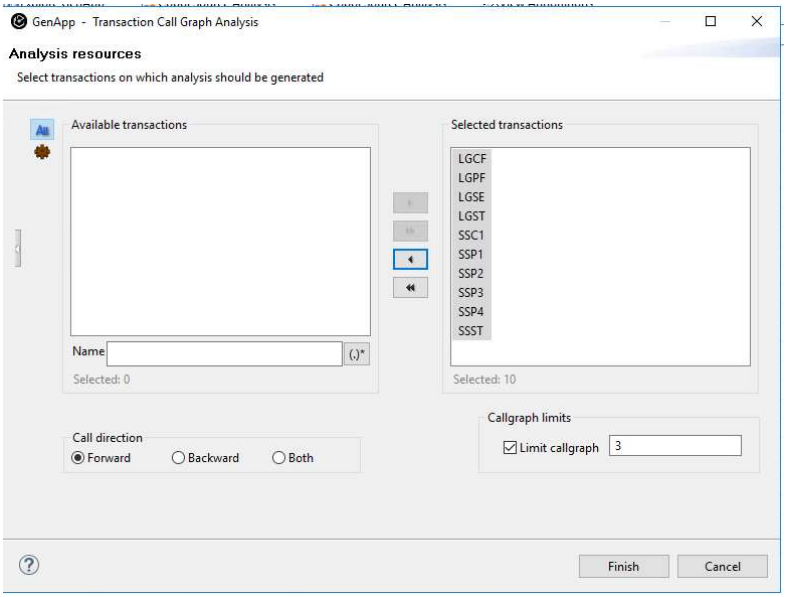
Now, let's focus on how the other transactions look like, to see if there is a pattern.

Compare with the other transactions

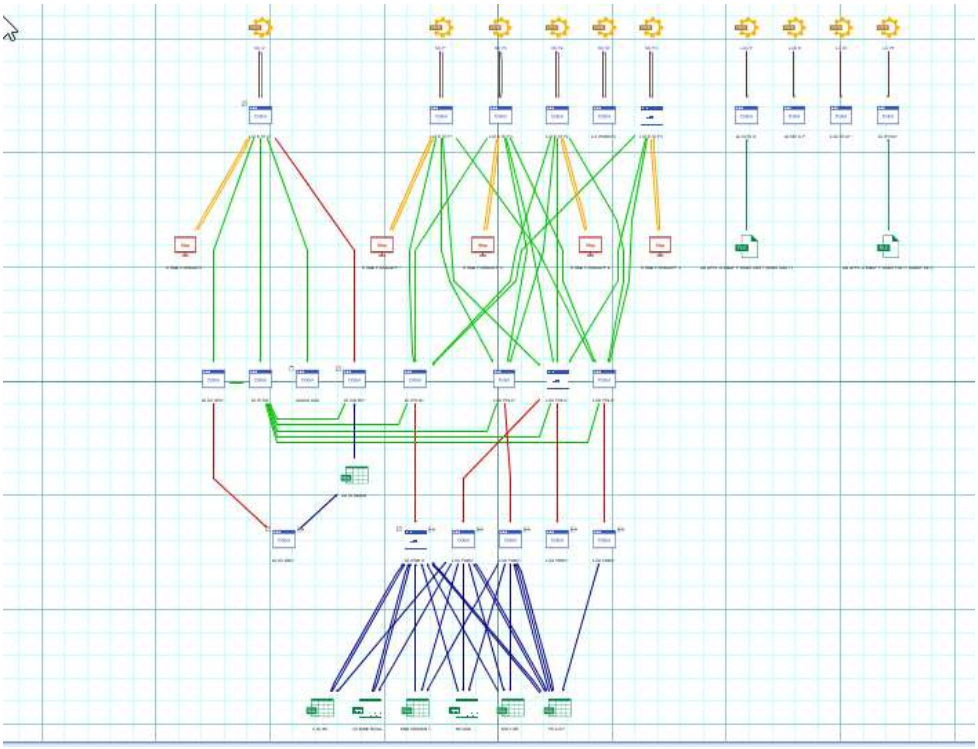
Go to the **Explore** tab, then go to the **Explore Projects** tab. Make sure GenApp is highlighted and select Transaction Callgraph.



Open the transaction call graph with all the transactions.



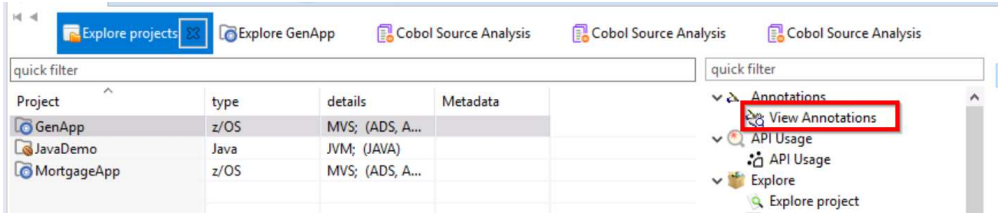
The following graph will be displayed:



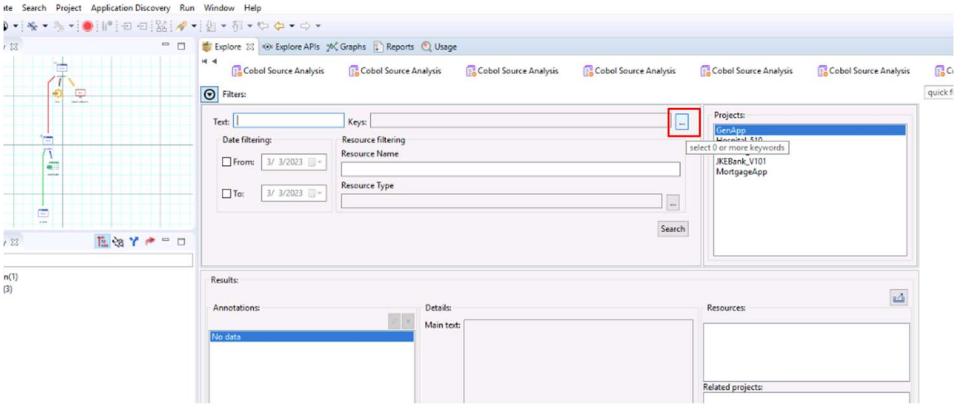
There is a mismatch with our program. It's not at the right level. There is a level missing. We need to get more details.

Retrieve the programs that we have annotated

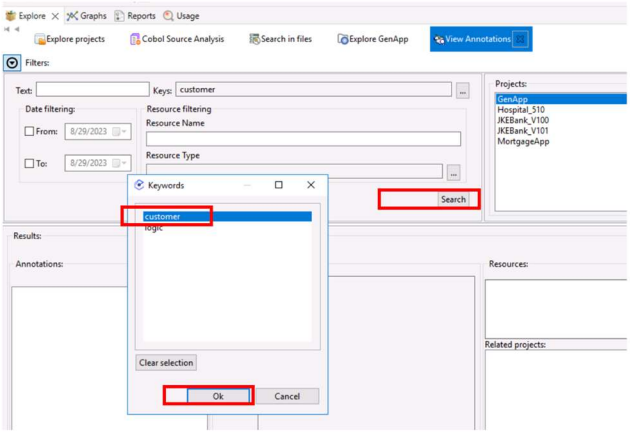
Before we continue our investigations, let's assume we are had a break and restarted the tool. We have forgotten our context. Let's query the annotations. Go to **Explore projects** tab, select GenApp, and click **View Annotations**.



We can see what keywords are already defined by clicking "…". We have Customer (as a keyword). Let's search with this keyword.

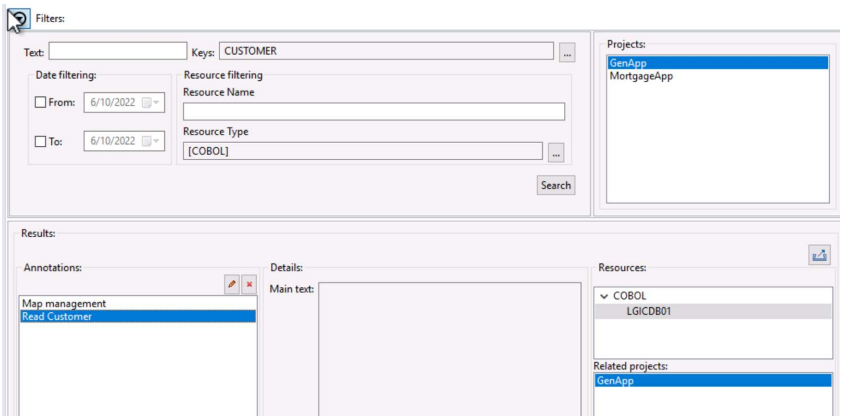


Select Cusomer, then OK, then Search



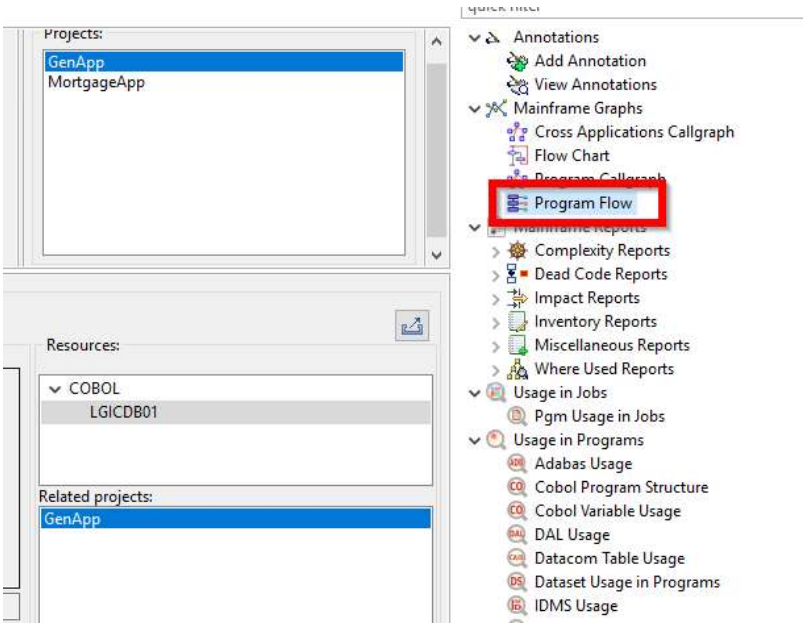
Now, we see our annotations. Annotations show us anyone's annotations added to this program based on their analysis.

(Note: hopefully more main text detail was added to explain why this annotation was made 😁)

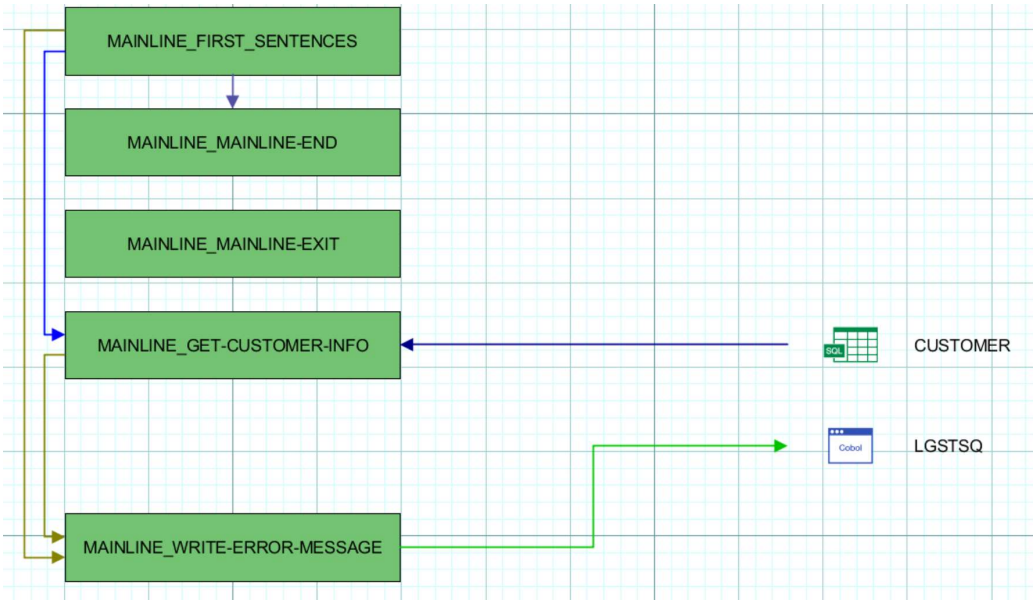


Investigate the Program flow

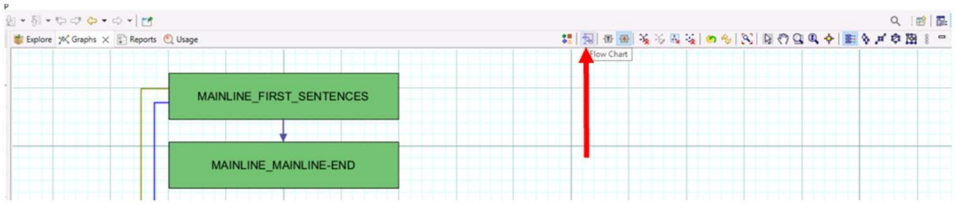
Let's first investigate what our database program is doing. We can do it from the annotation result or from the graph. Note the program name LGIDB01 needs to be selected



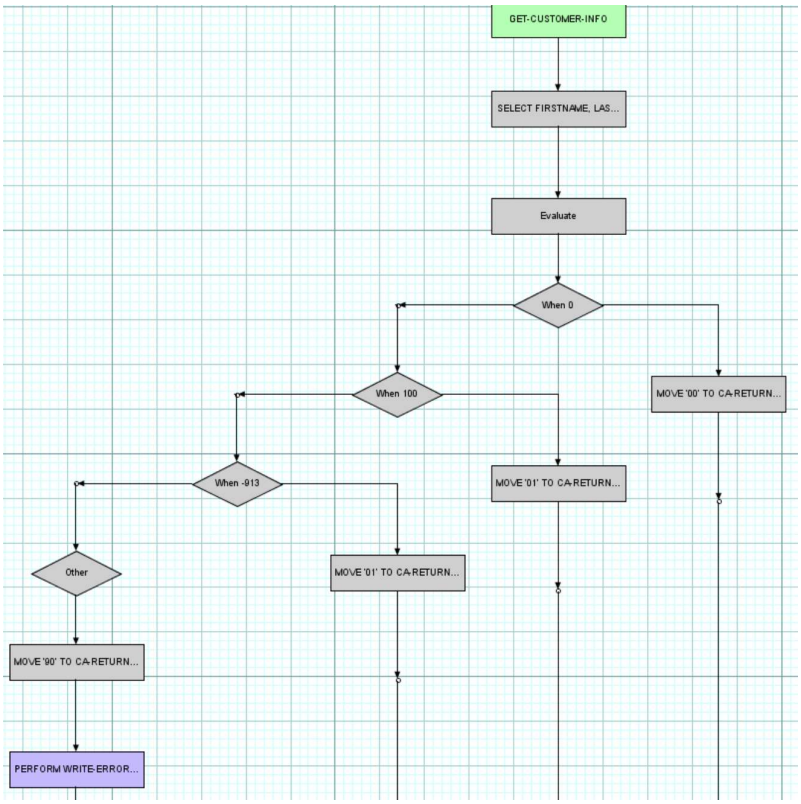
We see that the program reads the Customer details.



Explore the Flow chart of the program by clicking the  icon, located at the top of the screen.



This gives details about the way SQL Error are handled:

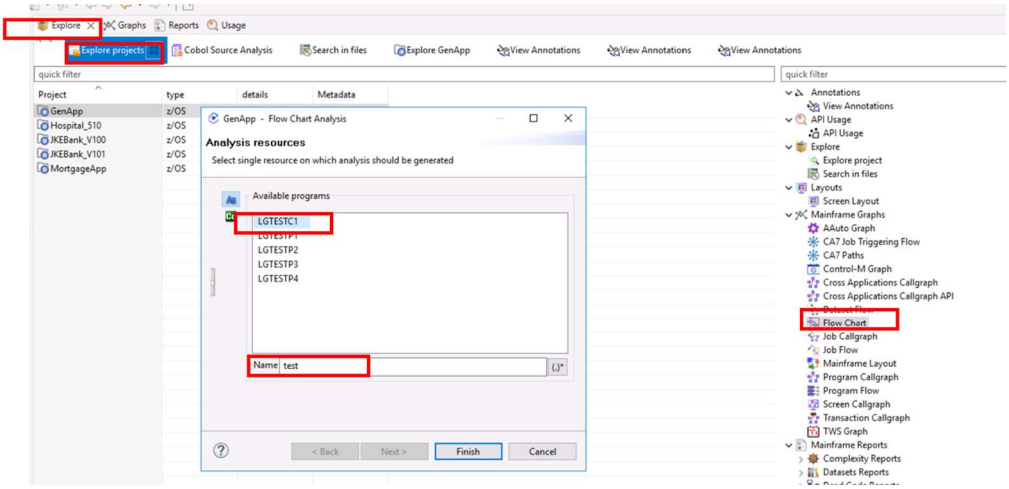


As you recall from the earlier analysis, program LGTESTC1 was also involved.

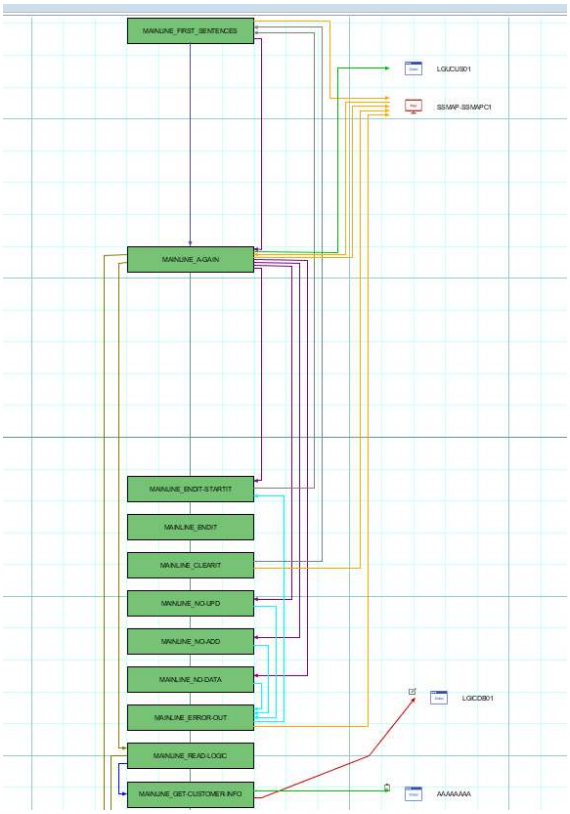
Let's open the program flow with the second program LGTESTC1.

Go back to the **Explore Projects** tab, make sure you are still in GenApp ,double click **Program Flow**,

Search for programs with “test” in the name, select LGTESTC1, click Finish



The following graph opens.



We see that it calls our database program but also has paragraphs for business logic (validation).

Partial conclusion

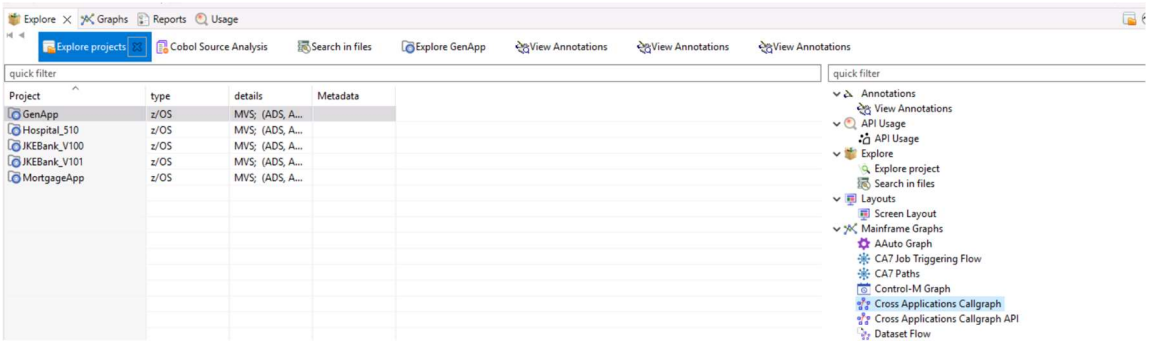
The map management and the retrieval of the data are handled in the same program. There is a pattern to separate these 2 processes. We need to introduce an intermediate level program, as done everywhere else.

It means that LGTESTC1 should not call directly LGICDB01. We need to introduce an intermediate program, using some contents from LGTESTC1. According to the rules, it should be called LGCUS01.

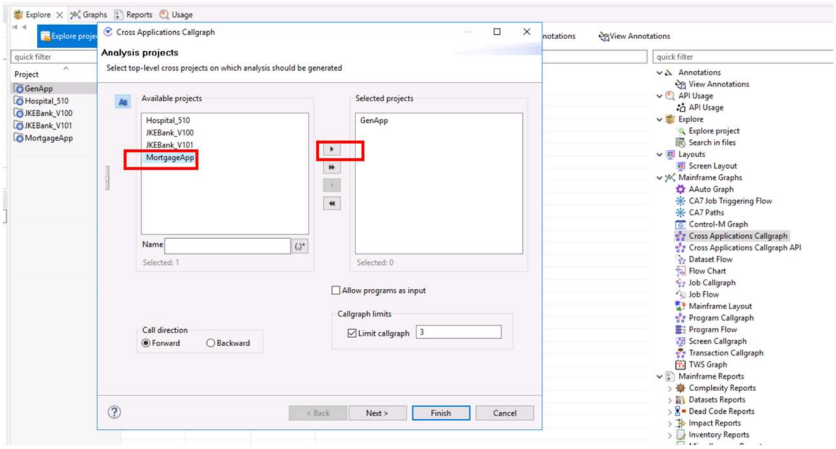
Explore Cross Application call graph

Before we start the refactoring, we need to check if there are adoptions to plan: Will the changes in GenApp impact other applications? Are our two programs used by another application? This is where Cross Application capabilities are involved.

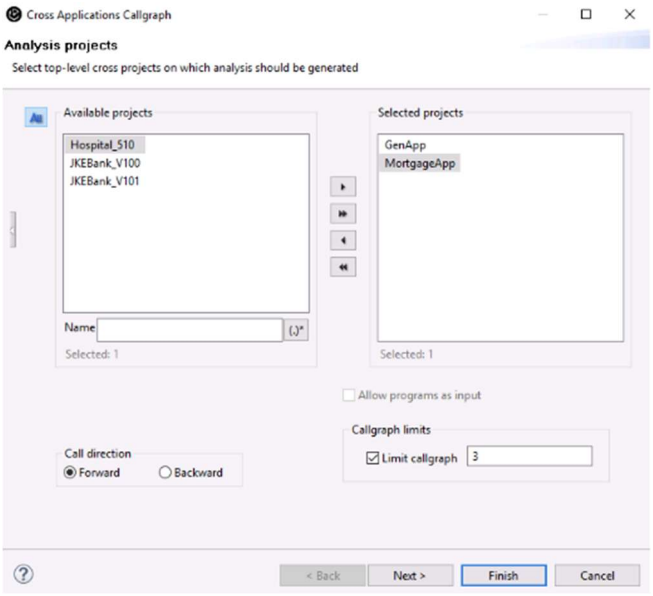
Let's perform a Cross Application Callgraph. Go back to the **Explore** tab and navigate to the **Explore Projects** tab. Select GenApp and click on Cross Application Callgraph.



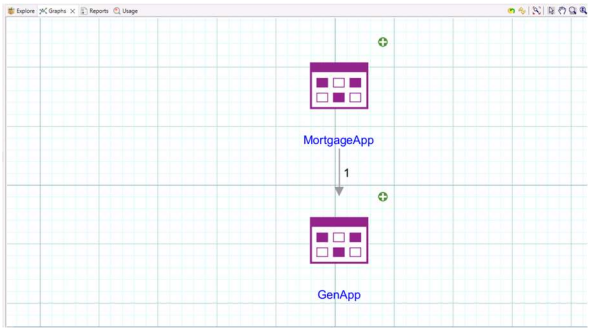
Add MortageApp to selected projects.



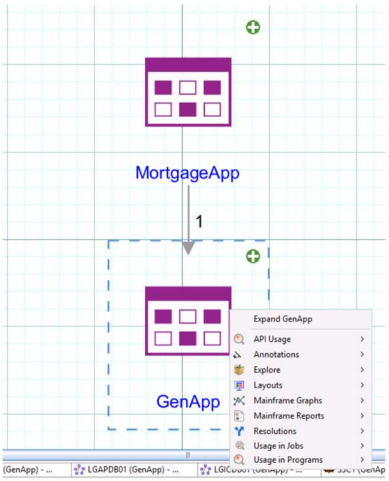
Click finish.



The following graph opens:



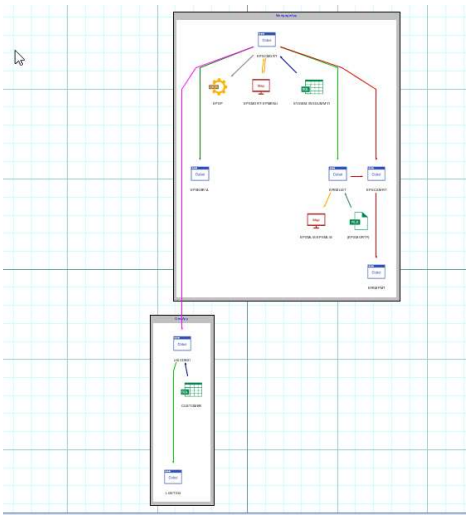
Select a project, then select Expand the Project (one at a time)



The following graph opens after you individually expand the projects

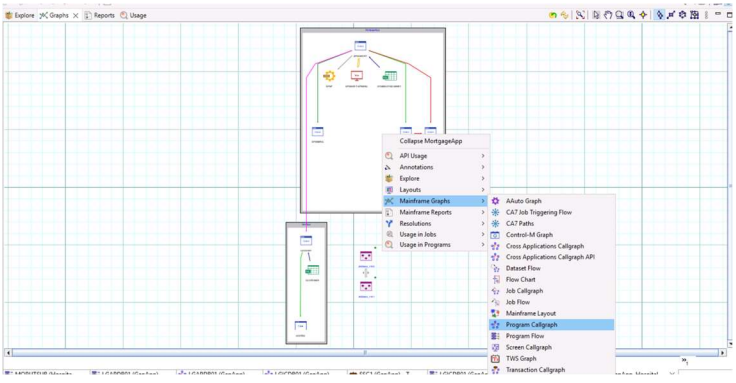
This only shows components “participating” in the interaction between projects

All other components are hidden. If you want to see all the other components of a project you will need to generate



The DB program LGICDB01 is used by another app. This app should adopt the new program (LGCUS01) that will contain the validation and logic. GenApp should not call directly LGICDB01, which is not meant to be exposed.

Note . If you want to see all the other components of a project, you will need to generate ProgramCall graph of the project.



Conclusion

We've completed the goal of this work shop to perform an application analysis to gain an understanding of what components and logic is involved in determining the scope of the original goal.

Goal: We want to assess the work to be done to expose an API that reads Customer information:

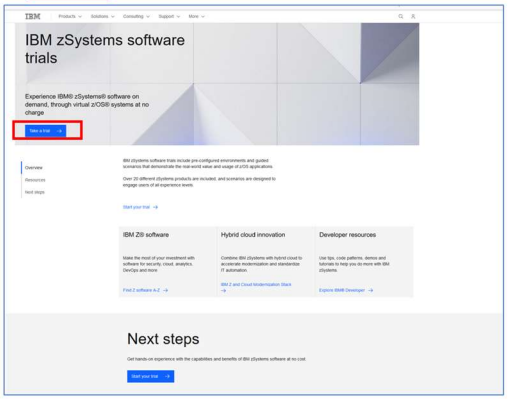
Now you have the information to be able to complete the refactoring.

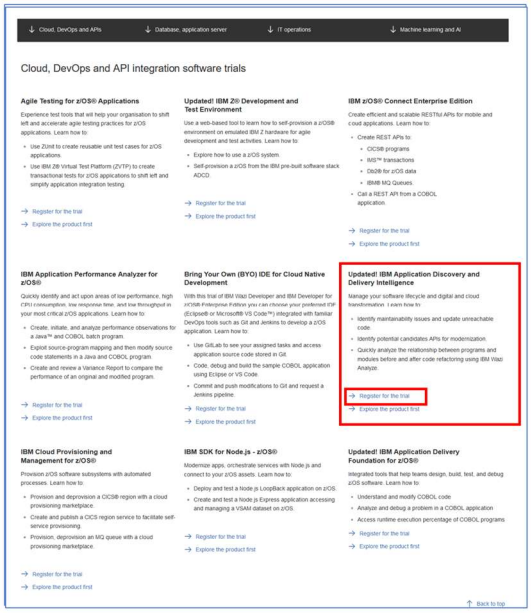
This is the work that needs to be done:

- Decompose LGTESTC1 and introduce LGCUS01 so that LGTESTC1 focuses on handling the map and LGCUS01 on the logic to obtain the data
- Expose LGCUS01 as a service. The data structure to use is in copybook LGCMAREA
- Adopt LGCUS01 in Mortgage Application

Note: The other scenarios on the on the zTrial instance would help in the remaining tasks.

You can request your own version of ADDI zTrial when you get back to your office and try them . https://www.ibm.com/z/trials

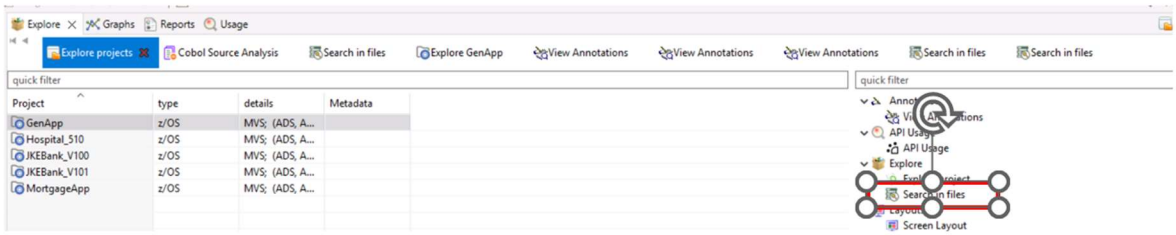




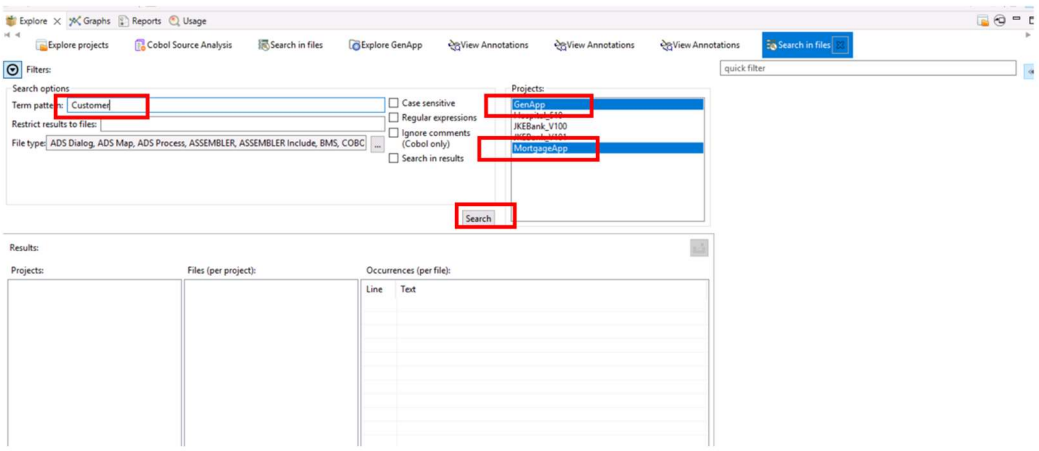
Optional - 3.14 (search)

We want to see where a term CUSTOMER shows up across/within Projects.

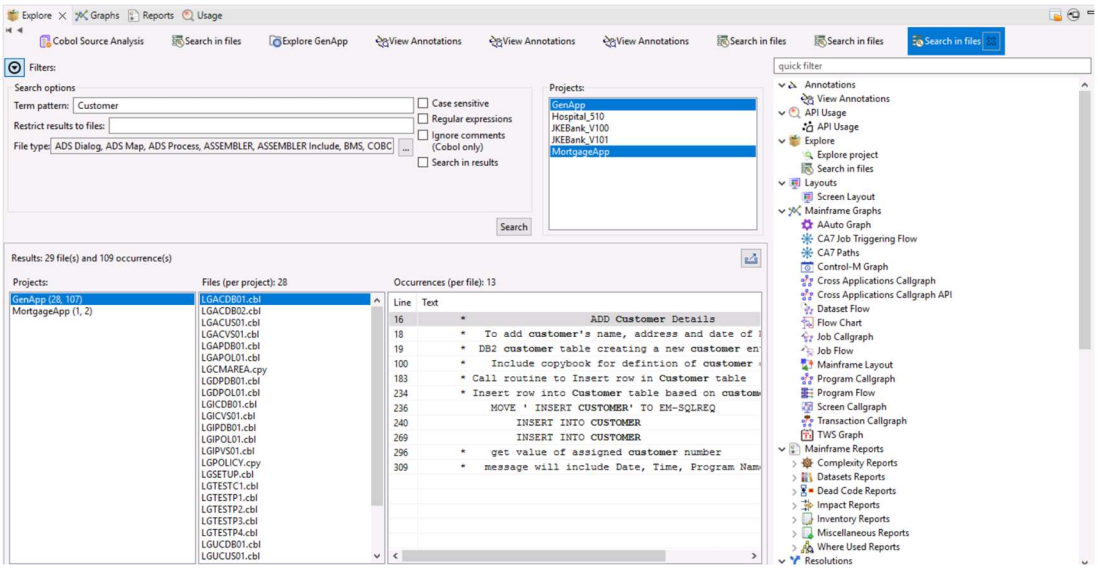
From the Explore Projects tab, select the GenApp project, and select **Search in files**:



Type Customer and select both the GenApp and MortgageApp, then click the search button



Let's see the results.



Search in ADDI is fast and can be performed across multiple projects. It's a simple entry point to exploration workflows. This result set tells you see all of the places that "Customer" exists