Remote Development

Development of a remote applications using Rational Developer for System z V7.5.

Lab Version V4.02

Last Updated: Monday, 10 August, 2009

Overview

This lab will show you how to develop applications running on a System z Mainframe system. You will define a remote z/OS connection, set up a MVS project, remote edit, compile and debug.

The lab can be divided into the following main tasks:

1. Initial preparation for RDz remote development
2. Working with remote programs
3. Compiling/linking/executing the remote program

Contents

Lab Details

Description	**Value**	**Comment**
	Local	
RDz Install path	C:\Program Files\IBM\SDP70	
Appserver Path	C:\Program Files\IBM\WebSphere\AppServer	
	Host general	
Userid	USER##	TSO user
Password	javaws	
Hostname / IP	192.168.7.xx	xx varies from 71 to 84
JES Port	9715	JES Job Monitor
MVS / USS Port	4039	Remote RSE Daemon
Remote Connection Profile	HostProfile	
User's home directory	/u/user##	USS home directory
User's HLQ	USER##	
	Prepared Workshop Files	

Dataset prefix for prepared code	ARNOLD	
Workshop files	/local/zoaws	

App Development Paths and Libs

Java home directories	/usr/lpp/java/J1.4 /usr/lpp/java/J1.5	Java 1.4.2 SR6 31 Bit Java 1.5.0 SR2 31 Bit
Link Libraries for COBOL Compile	SYS1.SCEELKED	
Debug Library HLQ	SYS1.DEBUG.V710	
Developer for z Library	SYS1.RD4Z.V750	

CICS Properties

CICS USS install directory	/usr/lpp/cicsts/cicsts32	
CICS user directory	/u/cicsts/cicsts32	
CICS var directory	/var/cicsts/	
CICS prefix	cicsts32	
CICS sit	CICSTS32.CICS1.SYSIN(CICS1)	
CICS Load Library	CICSTS32.CICS1.SAMPLE.LOAD	
Additional Link Library for CICS Compile	SYS1.CICS.V320.SDFHLOAD	
Catalog Manager	SYS1.CICS.V320.SDFHSAMP	
CICS Applid	CICS1	
CICS Debug port	9876	
CICS Web Service port	3602	
CICS ADM port	81	

Property Groups

	Batch	CICS	IMS
COBOL	batchCobol75	CICSCobol75	
PLI			
C/ C++			
Assembler			

General Hints: Everybody will get a different userid, distinguished by numbers. In your lab guide this individual number is represented by ##. Please always substitute ## by your number.

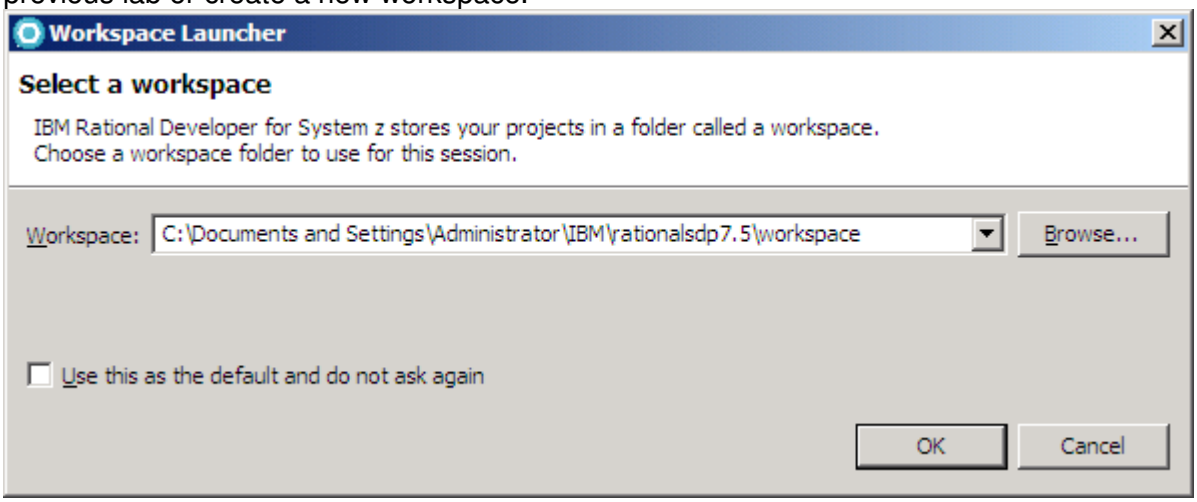
1 Initial preparation for RDz remote development

Before we can start programming some preparations have to be done.
First you will connect to a remote system which has been set up for you. You will learn how to organize your data using filters and how to allocate data sets using RDz. You will also use the workbench to submit JCL which will setup all required data for this workshop.

1.1 Define and connect to a remote system

The RSE's Remote Systems view shows all existing connections to remote systems. Connections are System Connection objects that are persisted, containing the information needed to access a particular remote host. The view contains a prompt to create new connections, and pop-up menu actions to rename, copy, delete, and reorder existing connections.
Connections contain attributes, or data, that is saved between sessions of the workbench. These attributes are the connection name, the remote system's host name and system type, an optional description, and a user Id that is used by default by each subordinate subsystem, at connection time. Underneath, all connections are stored as files in an Eclipse project named RemoteSystemsConnections, which the user can enable for team support, allowing connections to be shared by a team.

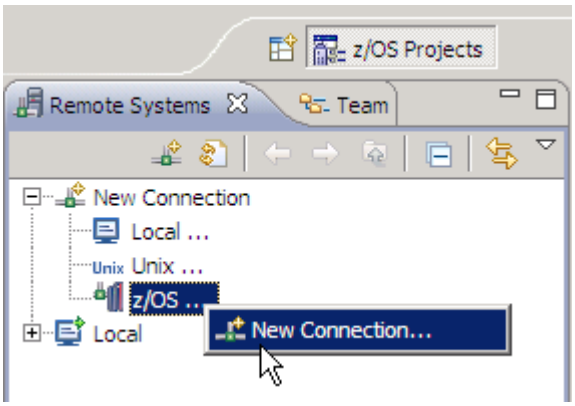
1. If Rational Developer for System z (RDz) is not already up and running, start it. When prompted for a workspace location you can either continue using a workspace from a previous lab or create a new workspace.



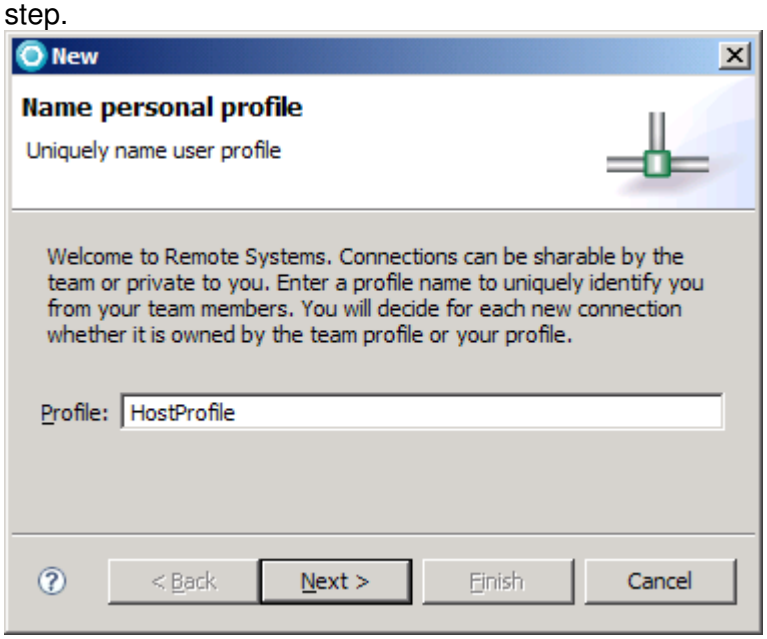
2. If you created a new workspace a welcome screen will be displayed that you will have to close.

3. Ensure you are working with the z/OS perspective by selecting Window → Open Perspective → Other… and select z/OS Projects.

4. Click on the Remote Systems View and expand the New Connection node, right-click on z/OS… and select New Connection to open the pop-up menu.

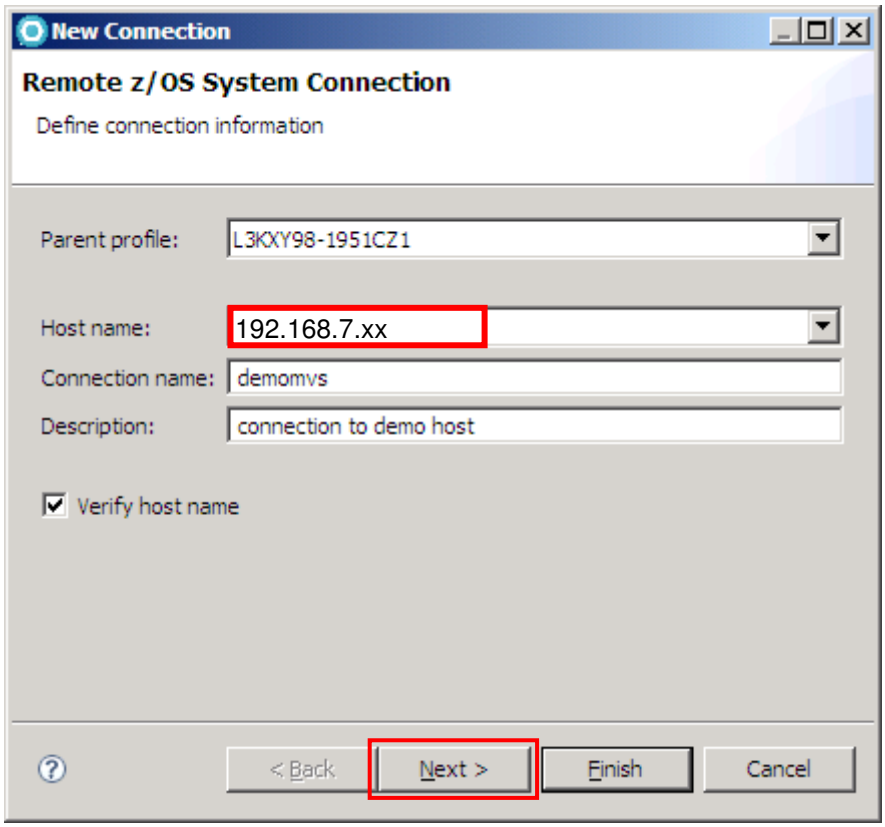


5. If this is the first time that you have attempted to create a connection in RSE, you are prompted to create a profile before you can create the new connection. Name your profile HostProfile and Click Next.
   If the screen does not appear there are already existing profiles and you can ignore this step.



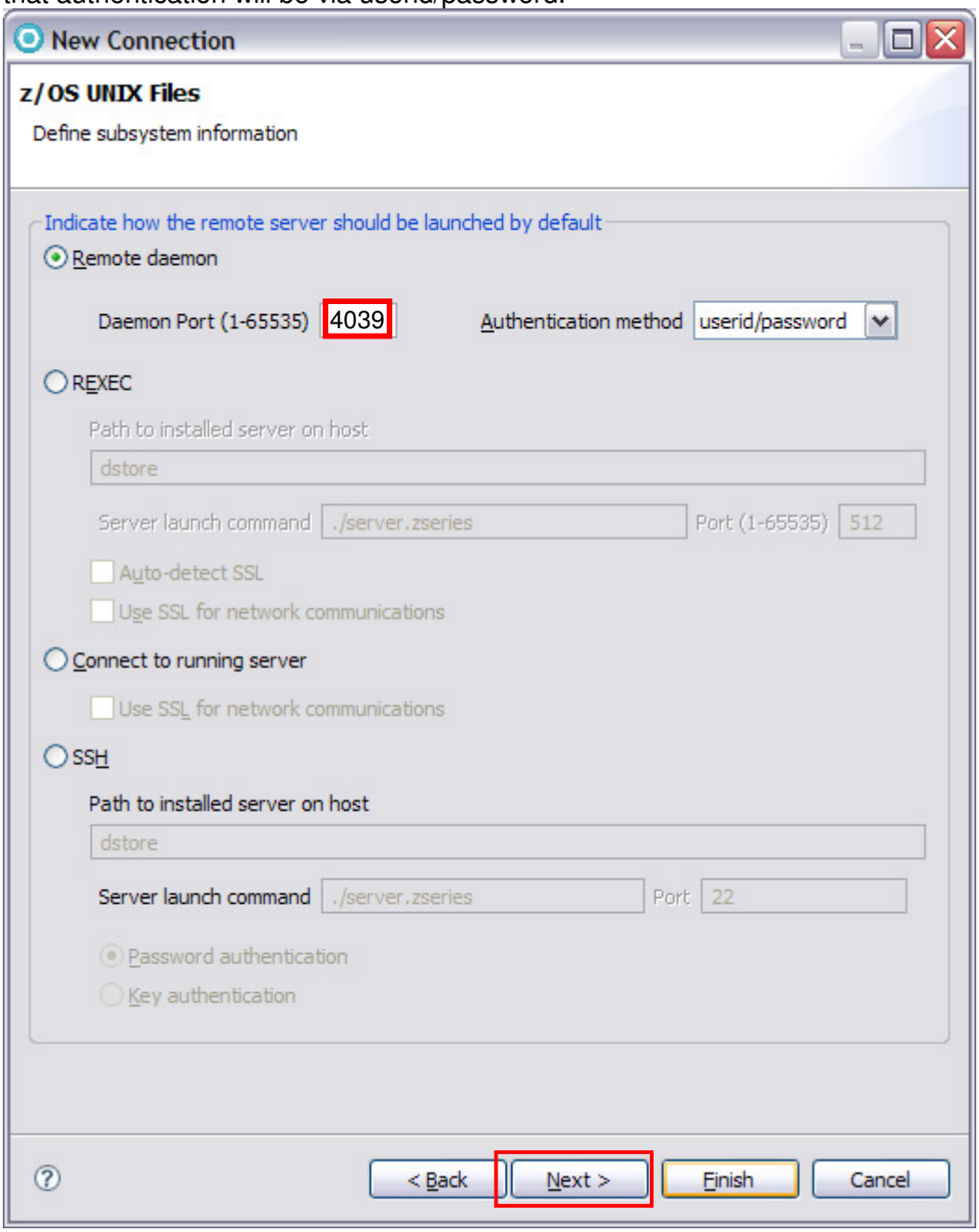
In the Parent profile field, the profile named after your workstation appears by default (Note that after you create the connection, you can share this profile to allow other users to have this connection in their RSE perspective).

6. In the Connection name field, type a unique name to identify your connection in the Remote Systems view called
demomvs.
The label that you assign to this connection will help you to differentiate between multiple connections to the same type of remote system.
In the Host name field, type
192.168.7.xx
as hostname or IP address of the z/OS system that you installed the RSE server on.
Optionally in the Description field, provide a short description of the z/OS system that you want to connect to.
To verify that the hostname or IP address in the Host name field is valid, select the Verify host name check box.

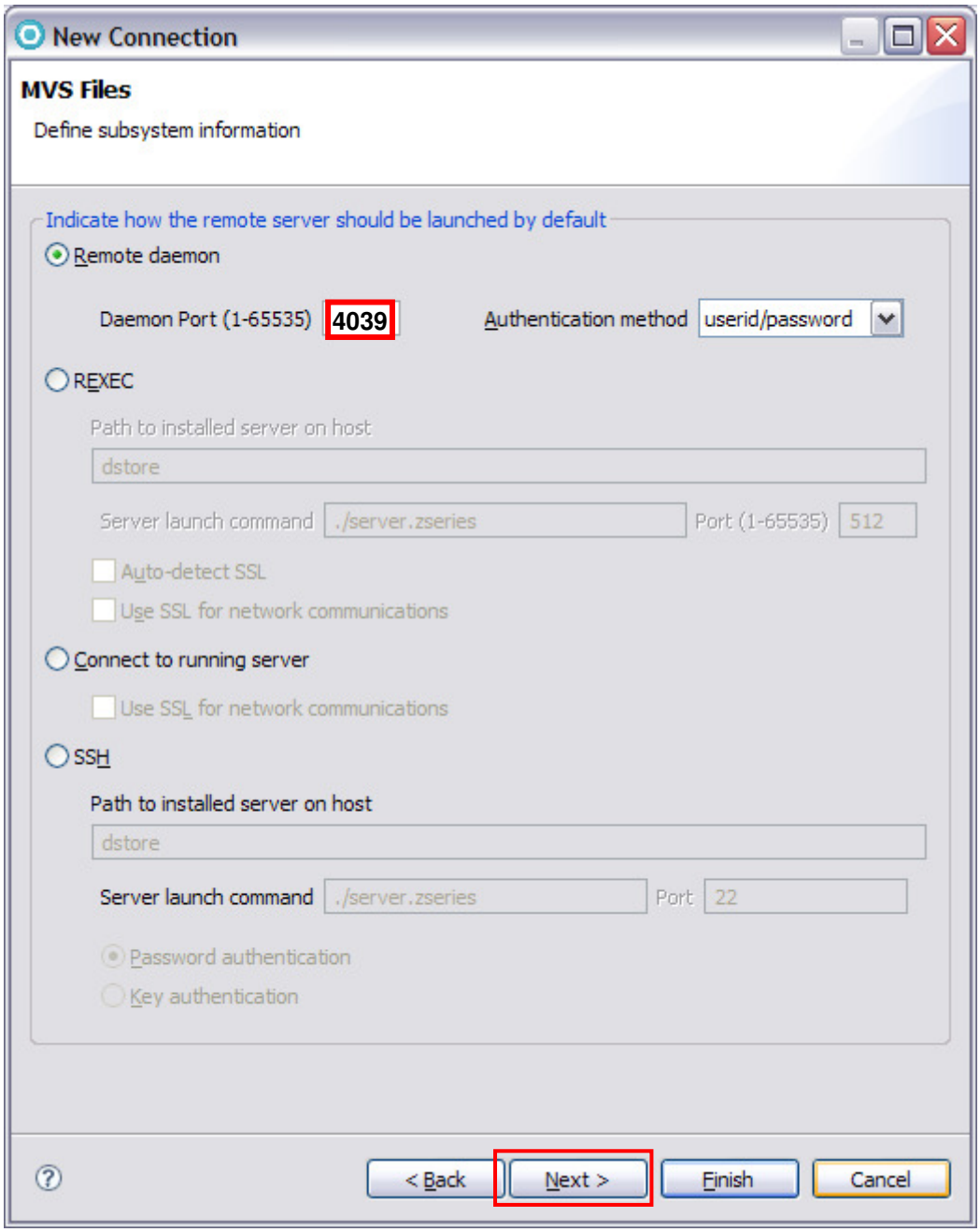


7. Click Next to continue.

8. In the USS Files window we will also use the Remote daemon. Use 4039 as port. Note that authentication will be via userid/password.



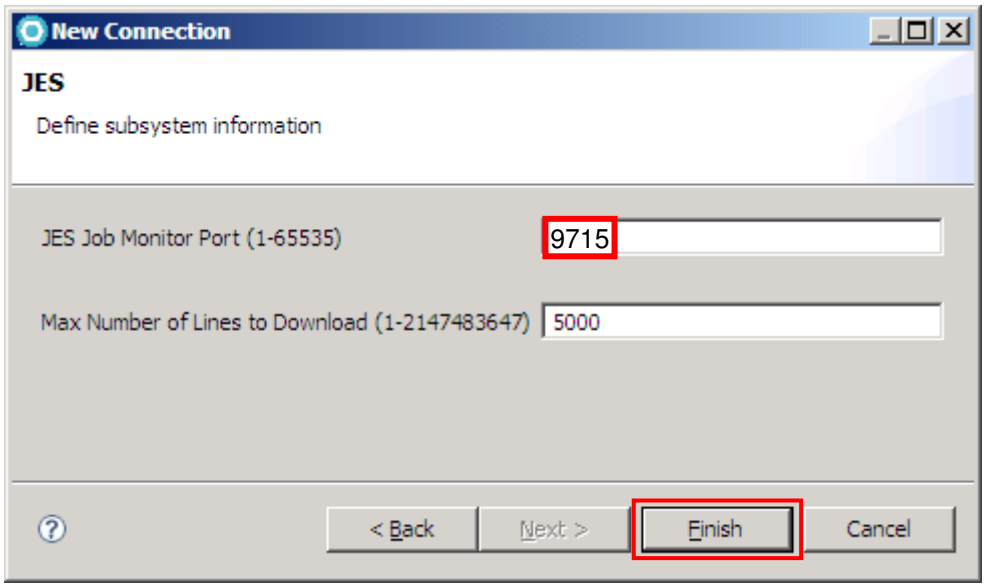
9. For the MVS files, enter **4039** as Remote Daemon port.



10. Click Next.

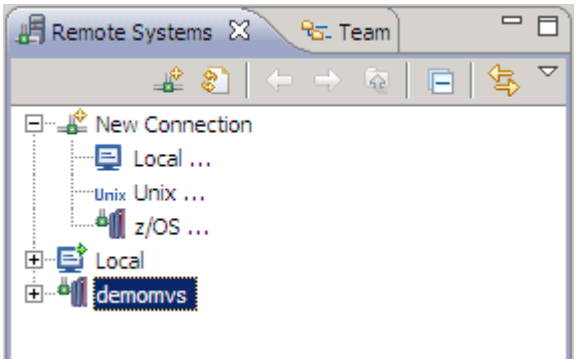
Note: If you have errors during the connection creation it is because the z/OS system name is not correct or not available (since you specified verify the host name on the step above).

11. In the JES Job Monitor Port field, insert
    9715
    as the port on which the Remote Job Monitor is listening.
    In the “Max Number of Lines to Download” field, type the number of lines to download before prompting you to specify if you want to download all of the lines in the dataset. We will accept the default of
    5000.

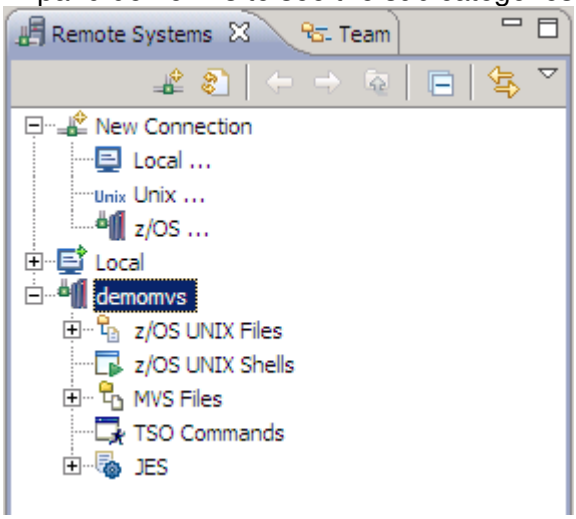


12. Click Finish to create the new z/OS connection and add it to the RSE perspective.

13. If network to the to z/OS is available, you will have the connection created as shown below:

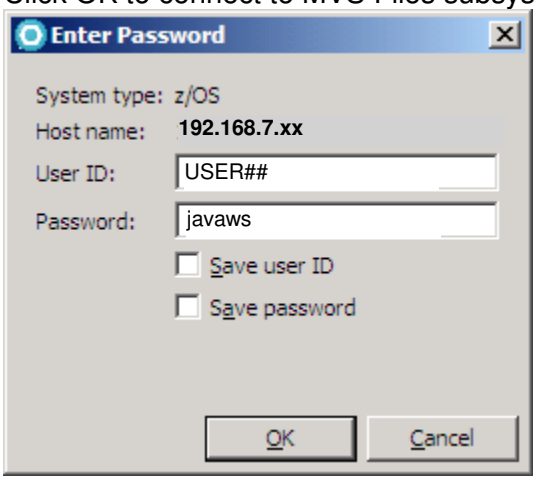


14. Expand demomvs to see the sub categories of your remote system.

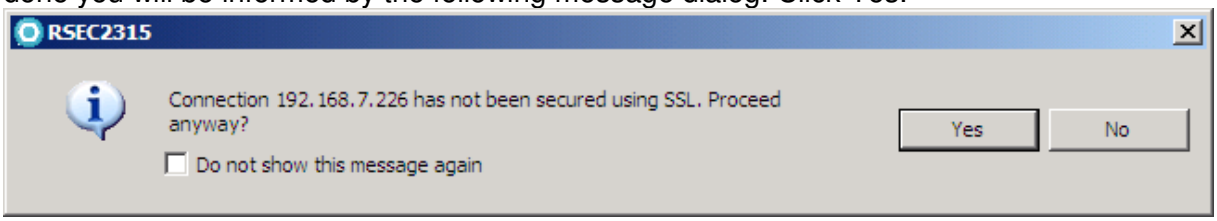


15. Now connect to your system by right-clicking demomvs and selecting “connect” from the context menu.

16. You will be prompted for your z/OS userid and password.
Type the assigned userid and password.
Click on Save user ID and Save password. You will be automatically connected at later times.
Click OK to connect to MVS Files subsystem.



17. Since RDz Version 7.5 you can secure your connection using SSL. If this has not been done you will be informed by the following message dialog. Click Yes.

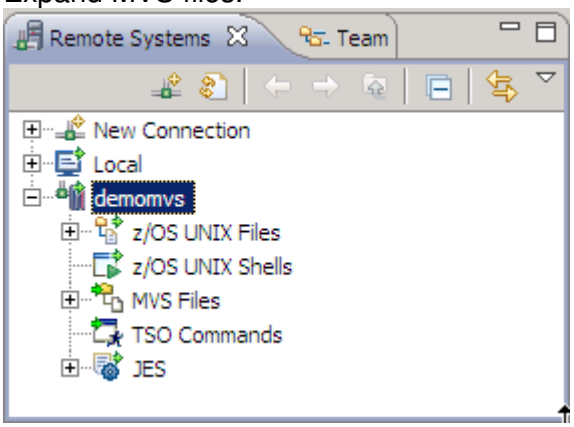


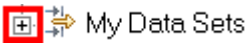
18. If you successfully connect to the remote system, the demomvs icon changes to demomvs (note the little green arrow).

1.2 Work with MVS files

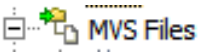
At this point you are ready to start working with the z/OS assets.
We will see how to allocate new PDS (Partition Data Set), copy members and use filters to display data.

19. Expand MVS files.

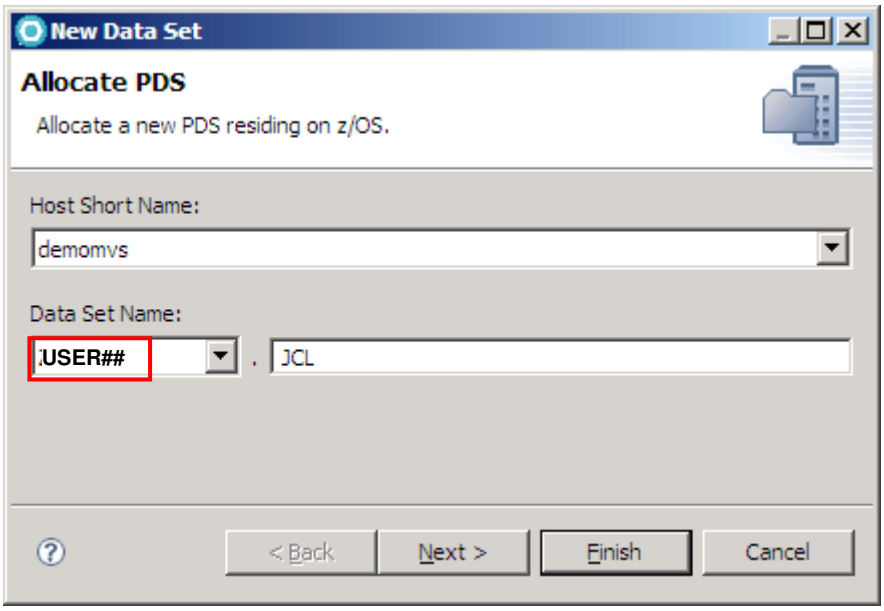


20. Click on  to see your data sets.

Don't worry if there are several data sets allocated, we will refresh the system later in this chapter.

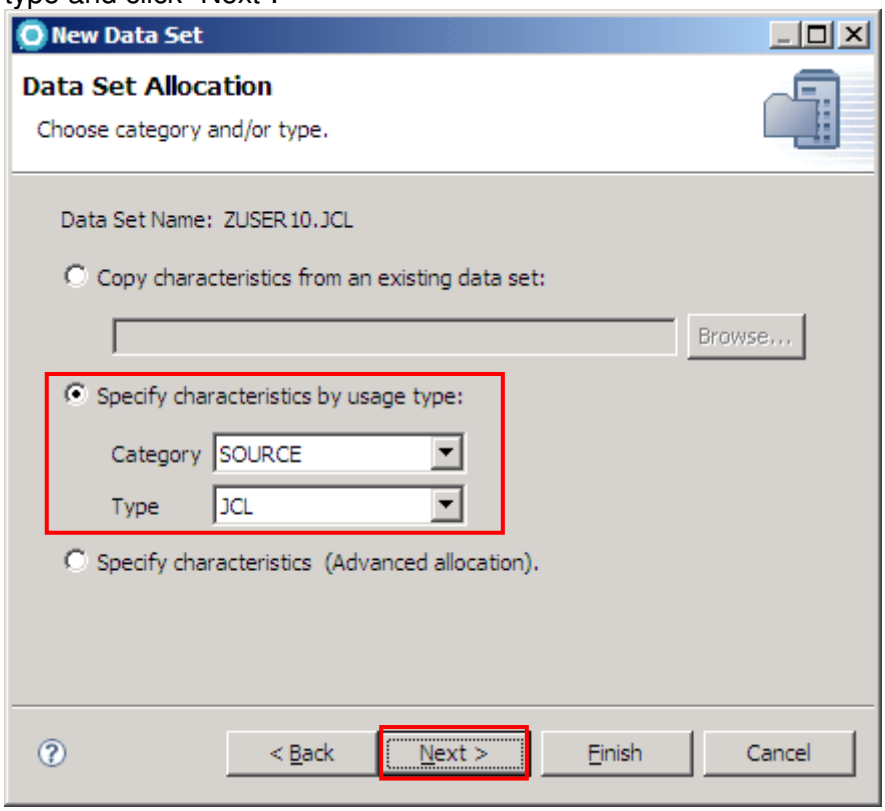
21. Right-click  and select "Allocate PDS" from the context menu.

22. Use the drop down list to select your High Level qualifier "USER##" and type the data set name JCL to allocate a data set that will contain our JCL members later on.



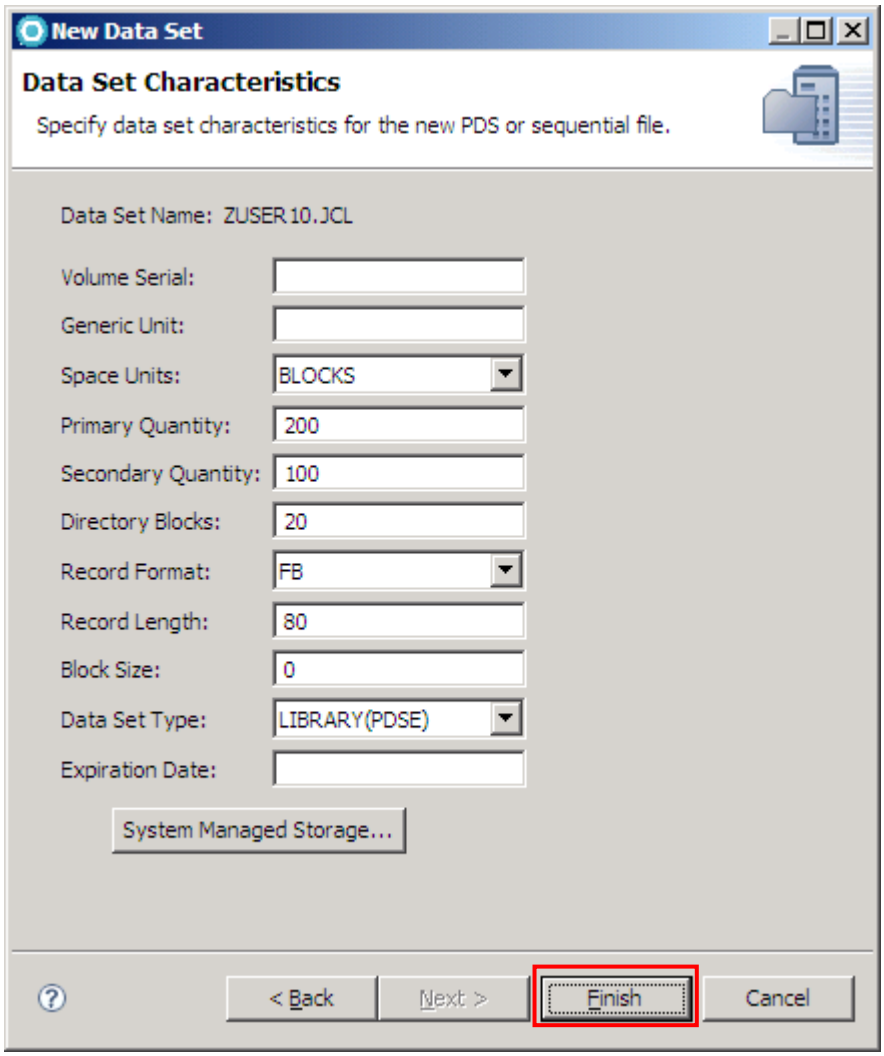
23. Click Next.

24. Select Specify characteristics by usage. Choose “SOURCE” for category and “JCL” for type and click “Next”.

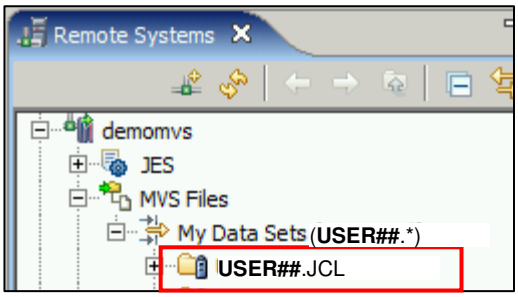


This is a nice feature since it allocates a data set that has the characteristics necessary for holding JCL.

25. Note the default values and click Finish.



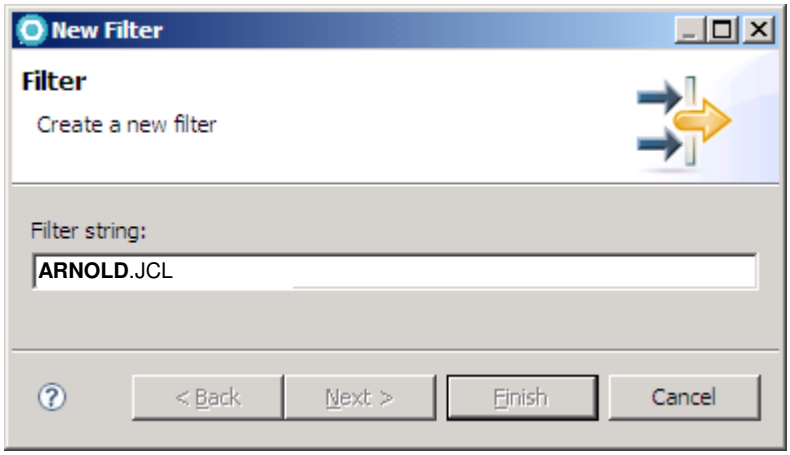
26. If space is available in the z/OS, this Data set will be created. If you do not see it, just right-click on MVS Files and select Refresh.



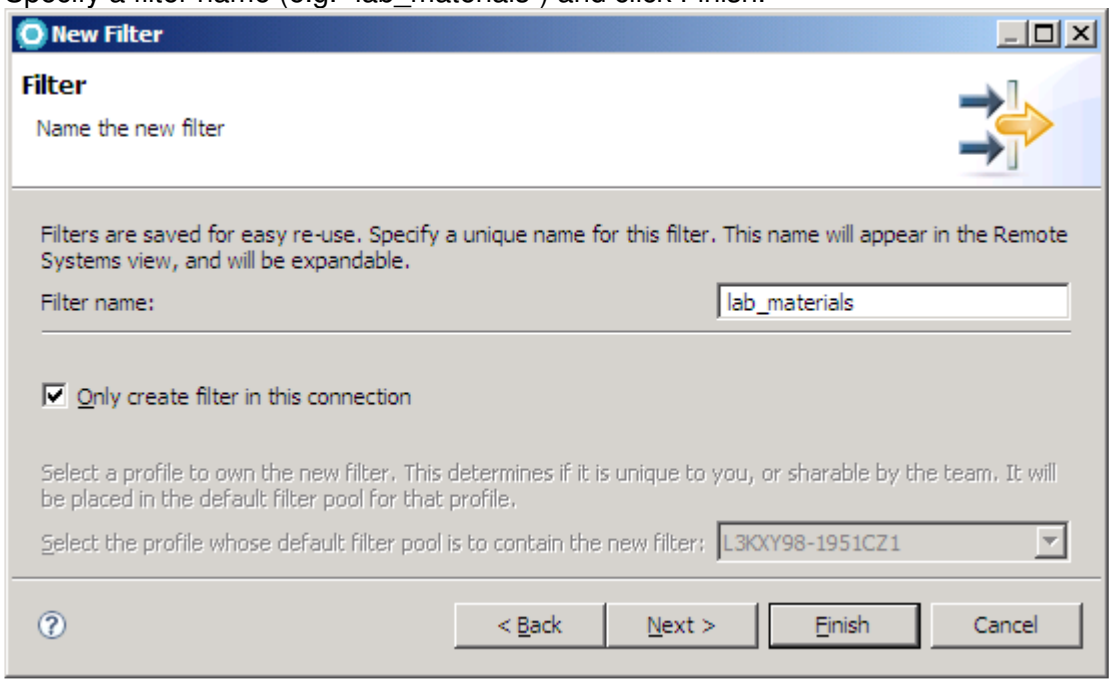
We will now create a filter to access our master data set ARNOLD.JCL and copy a member from there to our own JCL data set USER##.JCL. You already met a MVS filter when you expanded “My Data Sets” earlier in this lab. This is the default filter to display all data sets of the user currently connected to the system.

27. Right-click MVS-Files and select New → Filter from the context menu.

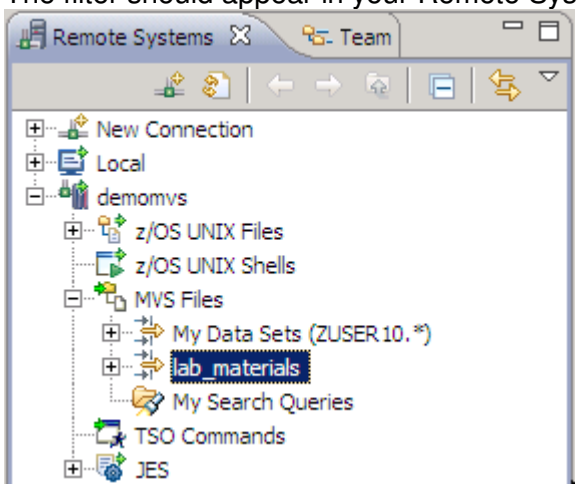
28. Specify “ARNOLD.JCL” as Filter string and click “Next”. Be aware that this is a master dataset, not your own. Please use exactly the name stated in the script here.



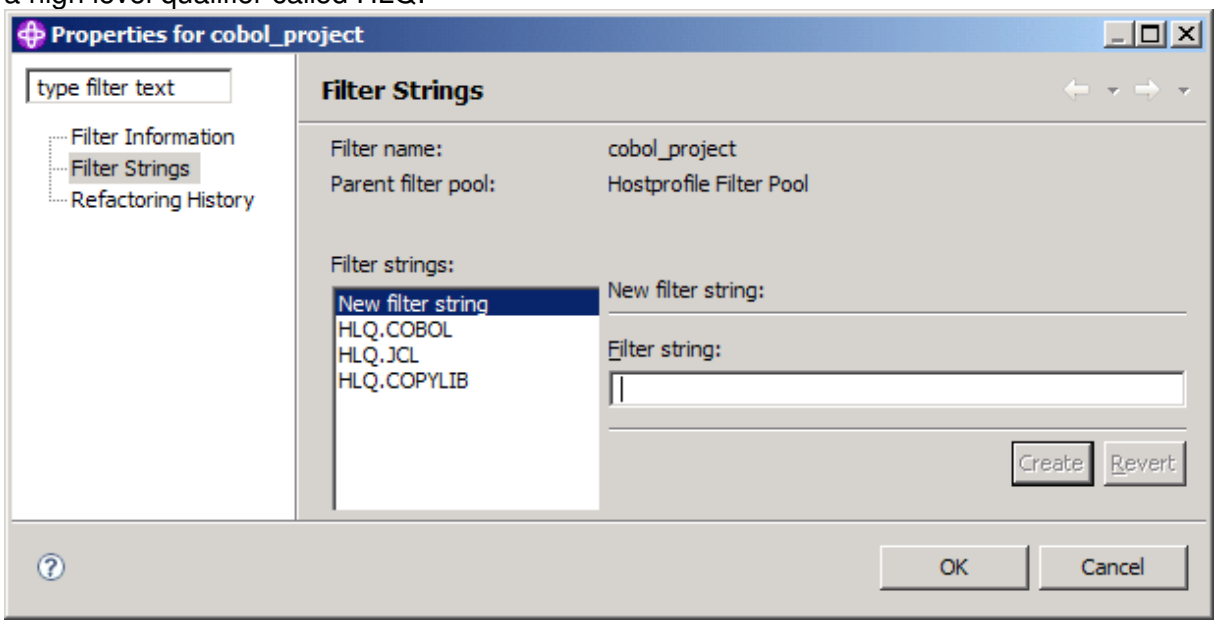
29. Specify a filter name (e.g. “lab_materials”) and click Finish.



The filter should appear in your Remote Systems View.



Note (you don't have to do this now), that after creation you could specify more filter criteria by right-clicking your filter and selecting Properties. You could create other filters like HLQ.*, HLQ.*.COBOL, HLQ.UTIL.*, HLQ.*.COB*, etc… (where HLQ is your high level qualifier)
Using this mechanism you can easily structure your workspace by for example displaying only the datasets required for a certain project using multiple filter criteria.
The following picture shows an example for a filter used in a COBOL programming project for a high level qualifier called HLQ:

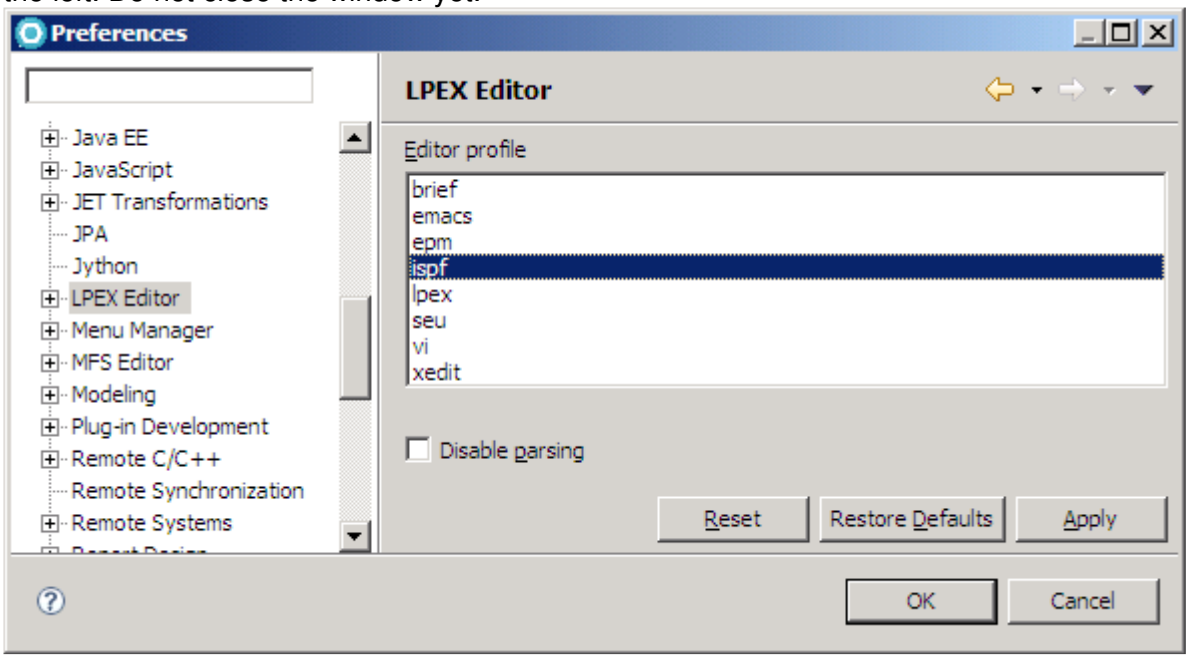


30. Expand the filter and navigate to the member RDZALLOC.jcl. Copy this member to your own JCL dataset you created.
    You can do this by right clicking the member and selecting Copy.
    Then right click USER##.JCL and click Paste.

1.3 Adjust the editor

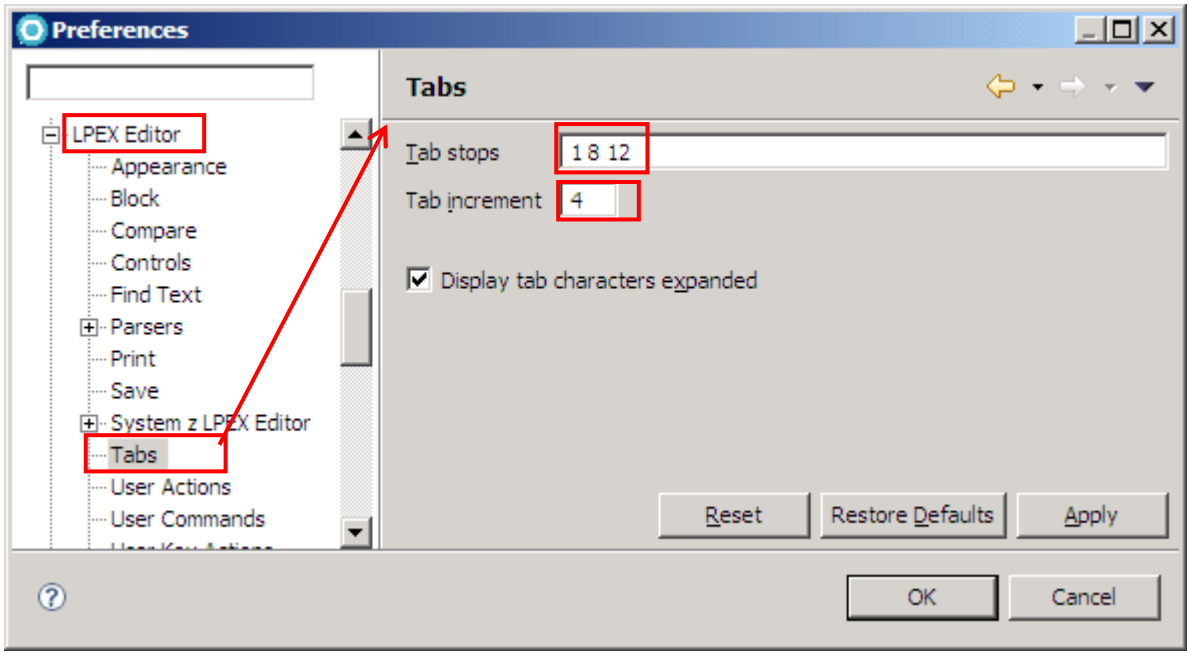
The default profile used by the LPEX editor is lpex, which defines the key behavior. The LPEX editor can emulate most of the key behavior of other editors like ISPF and XEDIT.

31. Open the Preferences dialog (Windows → Preferences) and select LPEX Editor from the left. Do not close the window yet.

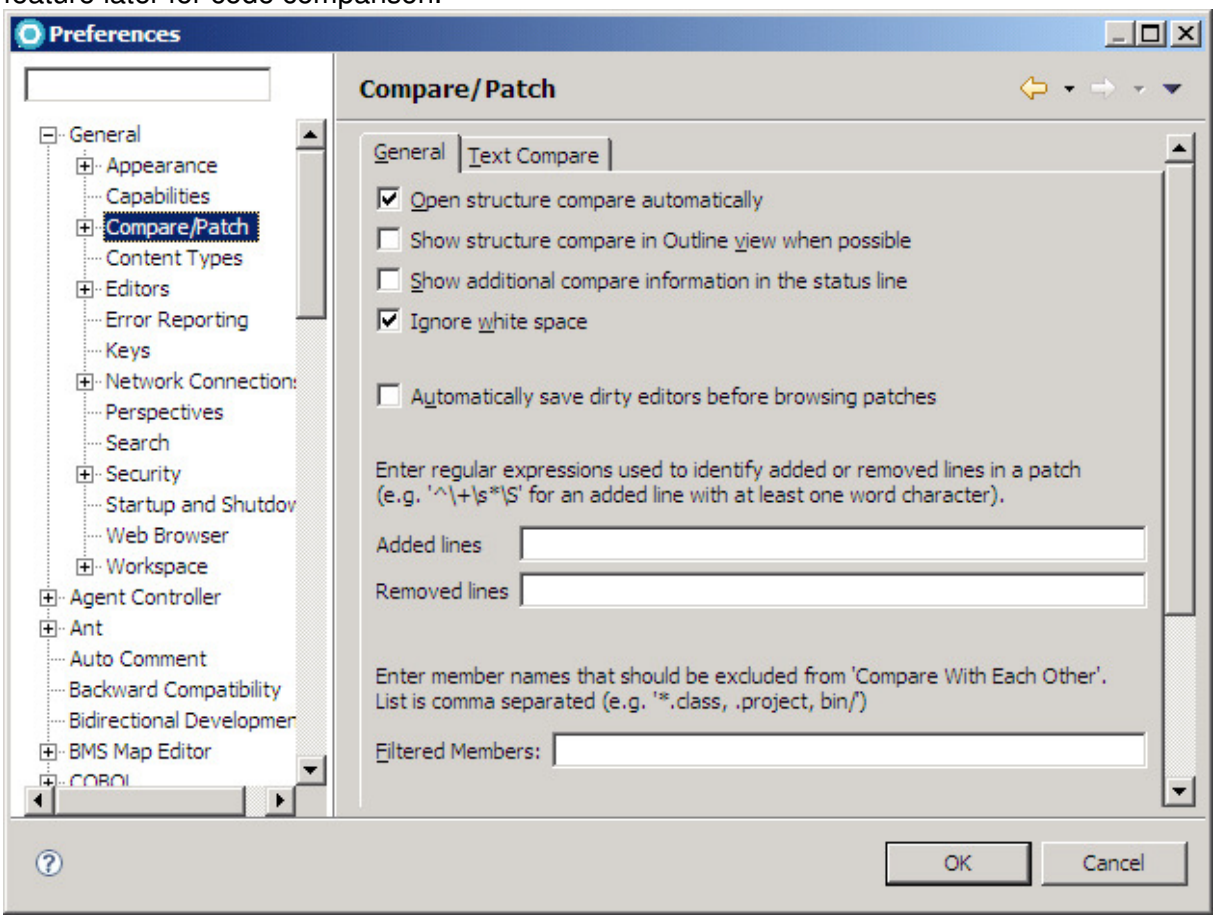


You now are able to enter prefix commands in the prefix area, such as d to delete, m to move, etc... like you ma be used to from ISPF.

32. Creating Tabs for the Editor
Since we are using COBOL is a good idea to create some TABS for easy navigation in the editor.
Expand LPEX Editor, click on Tabs and enter 1 8 12 as Tab stops. Also change the Tab increment to 4 (instead of 8). Each time Tab key is pressed will move the cursor to positions 1, 8 and 12. The increment 4 will also make stops at 14, 16, 20, and so on.



33. Go to General → Compare/ Patch and check “Ignore white spaces”. We need this feature later for code comparison.



34. Click Apply and then OK to save your changes.

1.4 Use JES (Job Entry System)

With RDz you can edit Job Control Language, submit jobs to JES and review the output all in one development environment.
We will now modify and submit JCL to z/OS to allocate some datasets that are required for this lab.

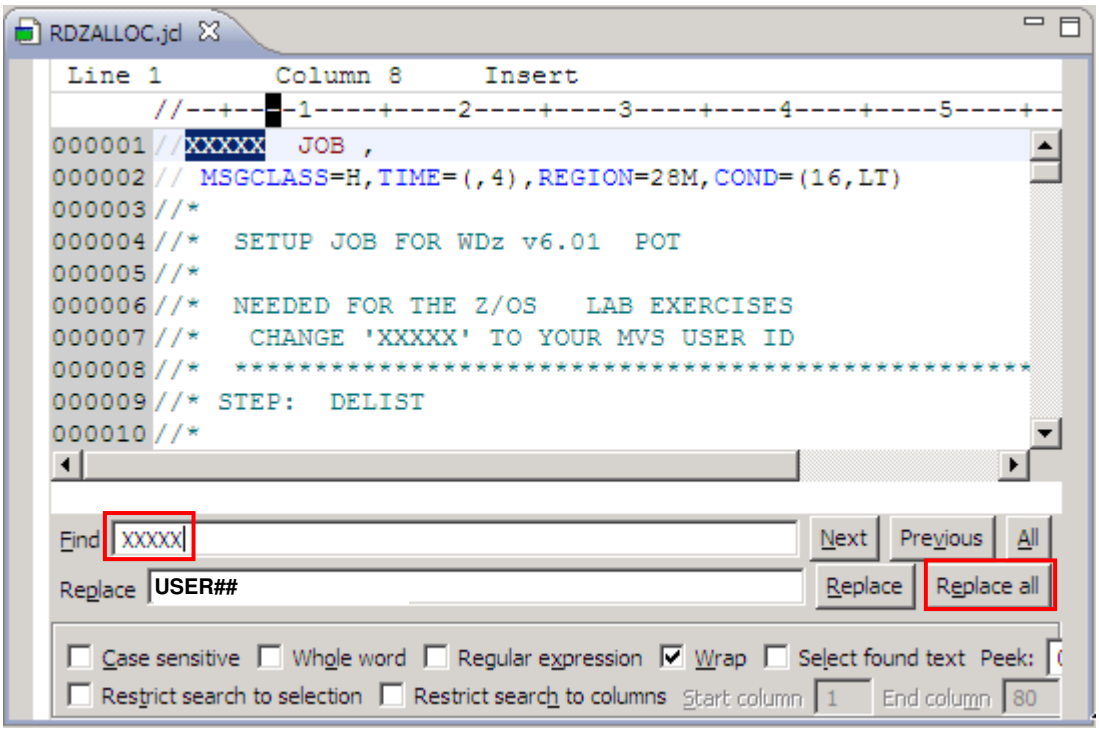
1.4.1 Explore some editor options

We will first use your JCL source to demonstrate some editor features. Those features apply to other sources than JCL as well, use them later if you find them appropriate.

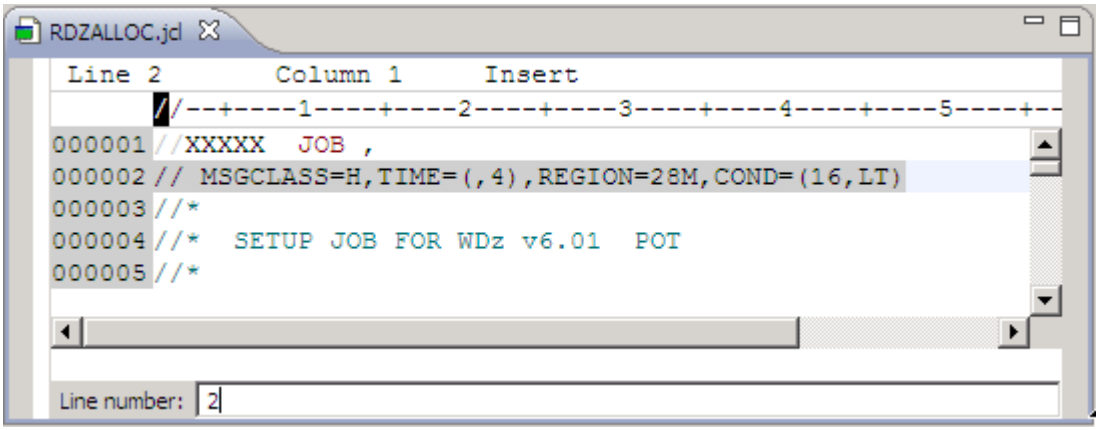
35. Double click on the copied RDZALLOC.jcl to open it.

36. You can use the mouse to expand the editor area to see more lines of the COBOL program. Also remember that when double clicking in the title (RDZALLOC.jcl) you can either make a full screen or return to original size. Also Window → Reset Perspective returns to the default.

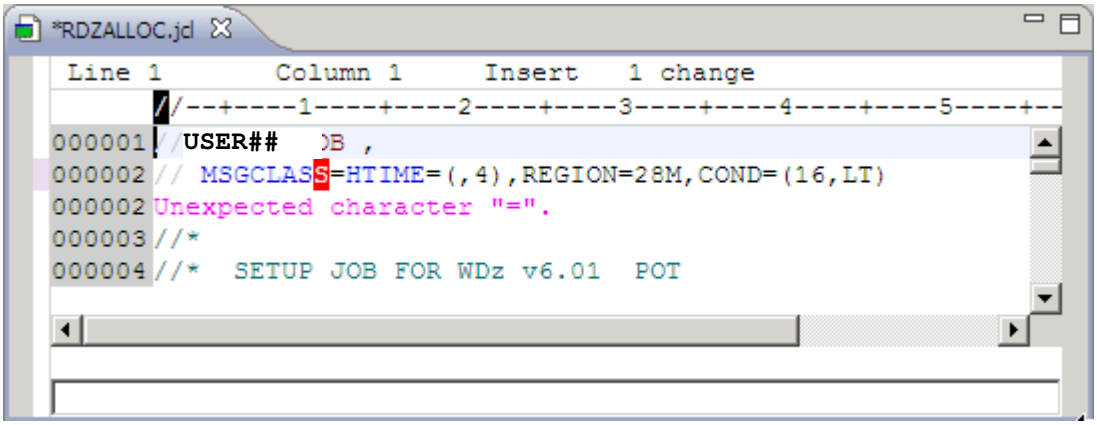
37. Change xxxxx to your z/OS userid (USER##) using Ctrl + F and clicking Replace all.



38. Hit Ctrl + L and enter 2 to navigate to line number 2.



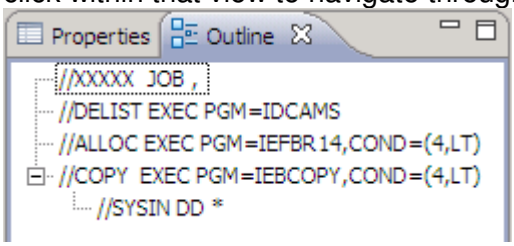
39. Add a JCL error, for example if you delete the first comma in this line and press enter this will have the error below.



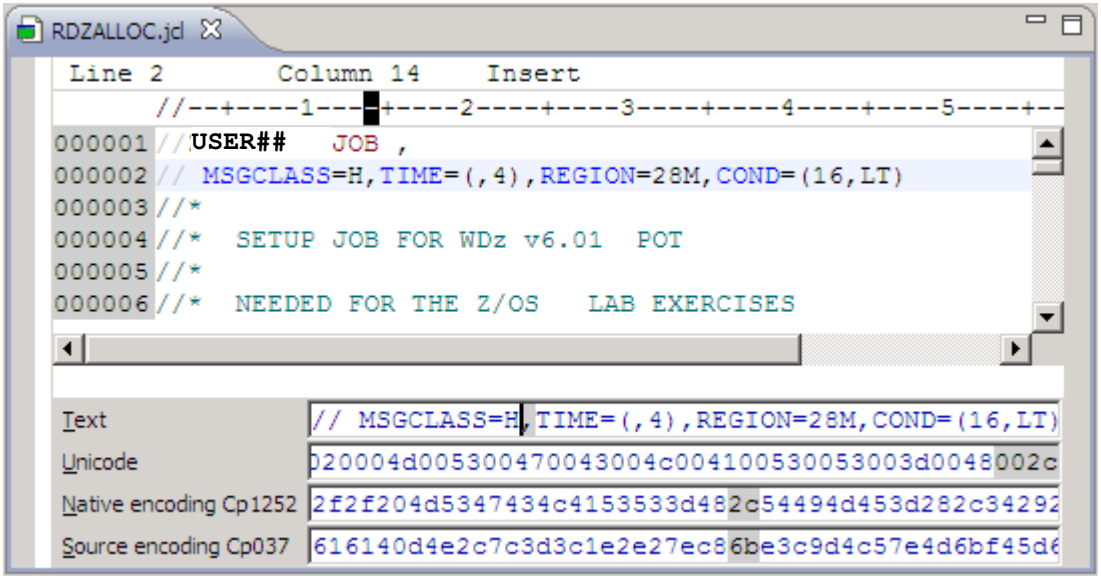
40. Fix the line.
    Either type the comma or hit Ctrl+Z to undo your change.

41. Save the changes (Ctrl + S). Note that the * that is beside of the title goes away.

42. Have a look at the Outline View where the structure of your code is displayed. You can click within that view to navigate through your code.



43. You could have a situation where you need to see hexadecimal contents of the fields (Common when doing assembler). Right-click and select Source → Hex edit line. You will have the result below:

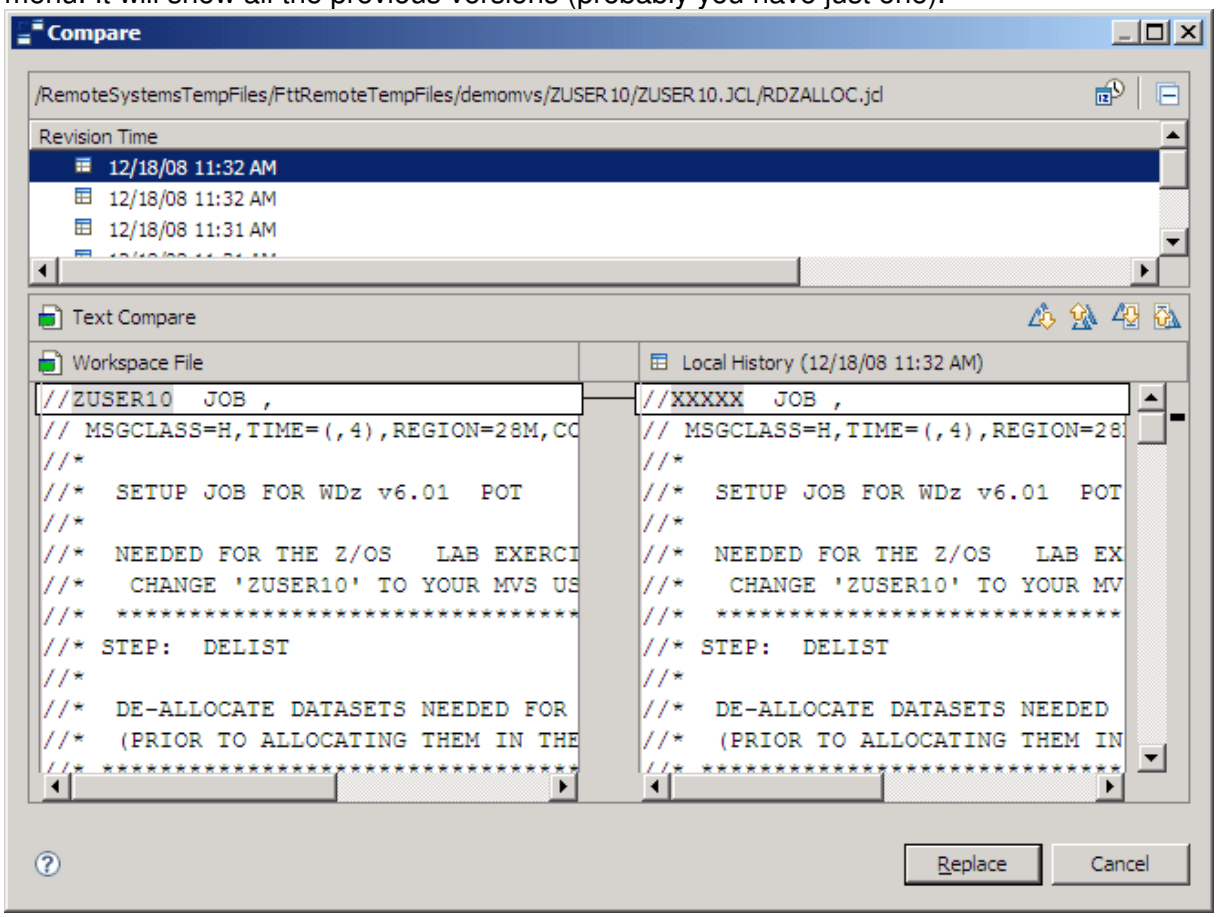


44. Right-Click in any place in the editor area and select View → Open new view. This is similar to the ISPF split screen. Sometimes this can be useful. Remember that double clicking in the title gives you a full screen and much more work space.

45. Right-Click again in any place in the editor area and select View → Horizontal split. This is another option. Also see other options like Next View and Previous view.

46. To return to the default view, use Right-Click inside of the second view and select View → Close View

47. One nice feature of RDz is the capability to recover previous versions using the local workstation. This is very useful when you delete and save components on the z/OS where undo is not possible.
Right click within the code and select “Replace with → Local History” from the context menu. It will show all the previous versions (probably you have just one).



48. As we do not want to undo our changes, press Esc to dismiss the dialog.

1.4.2 Submit JCL and review output

49. Submit this job by typing “sub” into the command line

```
RDZALLOC.jcl
Line 1        Column 14
      //--+----1----+----2-
000001 //USER##   JOB ,
000002 // MSGCLASS=H,TIME=(,
000003 //*
000004 //*  SETUP JOB FOR WD
000005 //*
000006 //*  NEEDED FOR THE Z
000007 //*   CHANGE 'USER##
000008 //*  ****************
000009 //* STEP:  DELIST
000010 //*
000011 //*  DE-ALLOCATE DATA
sub
```

50. Note that you could also right-click the member or within the editor and select “submit” from the context menu.

51. If the Job is not submitted and you get the message Job Monitor not connected to system as shown below, It is because your JES subsystem is not connected

```
RDZALLOC.jcl
Line 8        Column 48     Insert
      //--+----1----+----2----+----3----+----4--
000001 //USER##   JOB ,
000002 // MSGCLASS=H,TIME=(,4),REGION=28M,COND=(1
000003 //*
000004 //*  SETUP JOB FOR WDz v6.01  POT
000005 //*
000006 //*  NEEDED FOR THE Z/OS   LAB EXERCISES
000007 //*   CHANGE 'USER01' TO YOUR MVS USER ID
000008 //*  ************************************
The Job Monitor server is not connected.
sub
```

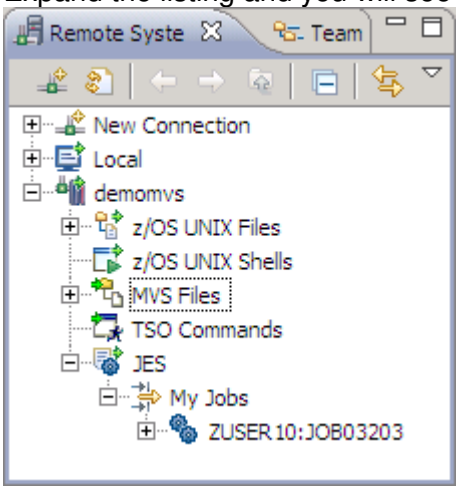
52. Use the Remote Systems view, locate the note JES under demomvs and Right-click on JES and select Connect.
    Try to submit again.

53. If submit succeeded you will see the JOB ID that was created for this execution

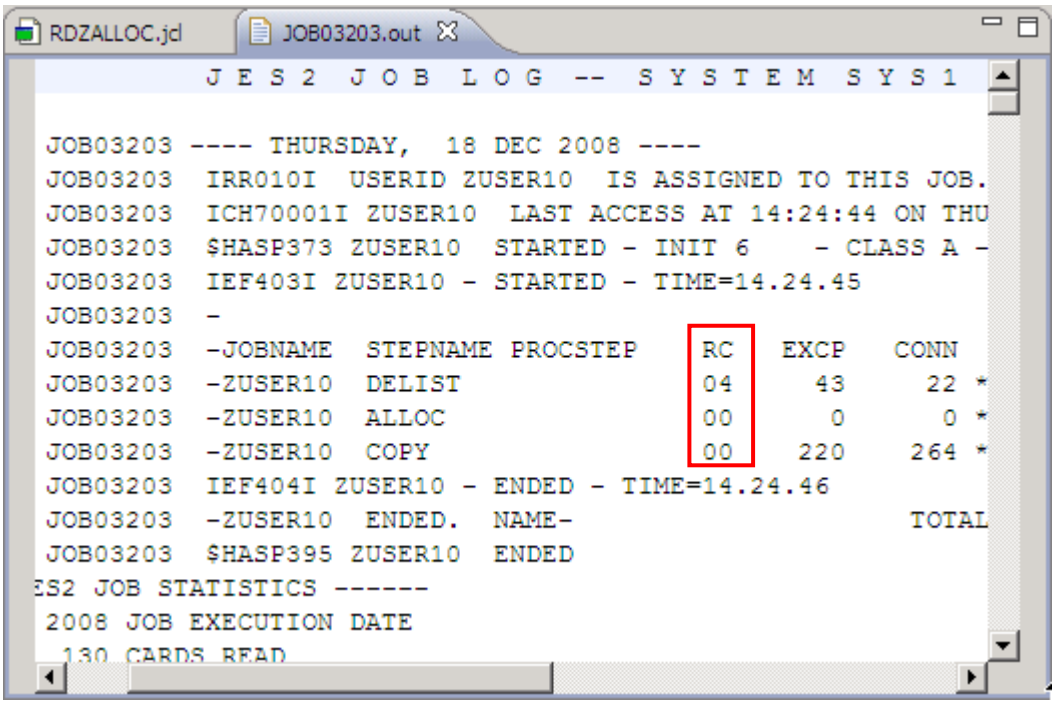
```
RDZALLOC.jcl
Line 1        Column 14     Insert
      //--+----1----+----2----+----3----+----4----+----5----+--
000001 //USER##   JOB ,
000002 // MSGCLASS=H,TIME=(,4),REGION=28M,COND=(16,LT)
000003 //*
000004 //*  SETUP JOB FOR WDz v6.01  POT
000005 //*
000006 //*  NEEDED FOR THE Z/OS   LAB EXERCISES
000007 //*   CHANGE 'USER##  ' TO YOUR MVS USER ID
000008 //*  ***********************************************
000009 //* STEP:  DELIST
000010 //*
000011 //*  DE-ALLOCATE DATASETS NEEDED FOR POT
JOBID: JOB03203
```

54. Go back to the Remote Systems view and under JES node (must expand demomvs), right click on the “My Jobs” filter and select Refresh (You can also select the filter and hit the F5 key

55. Expand the listing and you will see something like:



56. Double click on first listing and be sure that the job has successfully executed and has the return code is 4, 0 and 0 as shown below: (note could all zeros if you had datasets allocated already). If you receive higher return codes contact your lab instructor.

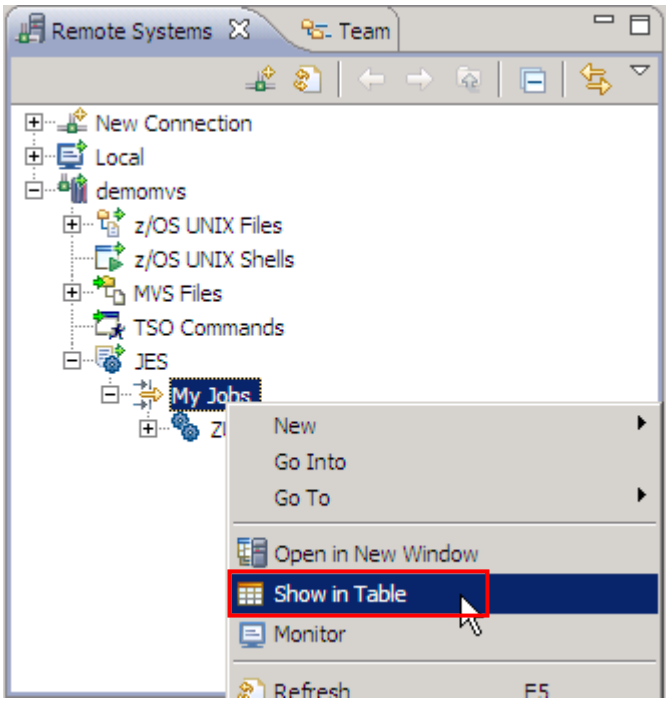


You just submitted a JOB that was executed in the z/OS and allocated some data sets that you will need for this lab.

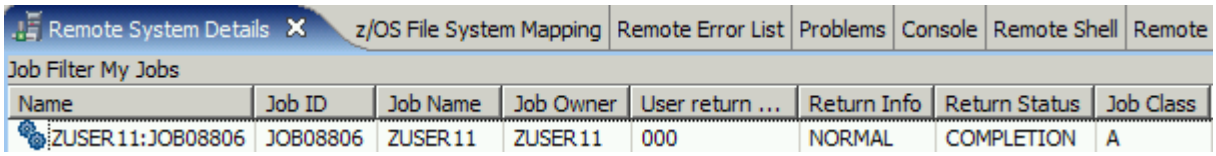
57. Close the editors that are opened and the JES2 JOB LOG.

To display your submitted jobs in a more “SDSF-like” format, you can do the following:

58. Right-click either the JES icon or any job filter and select “Show in Table” from the context menu.



59. The “Remote System Details” View displays your Jobs and corresponding Information in a table.



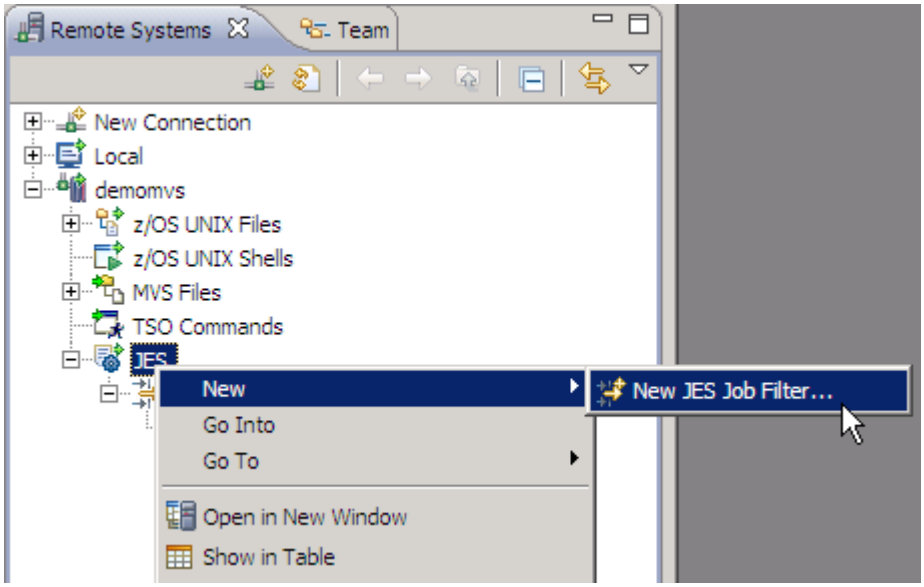
60. You can purge the output of jobs listed.
    To purge Go to the Remote System view, Right-click on your Job ID and select Purge. Alternatively you can purge the job as well from the Remote Systems Details table.

61. Using the Remote Systems view, right click on your “My Data Sets” filter and select Refresh (alternatively select the filter and hit F5).

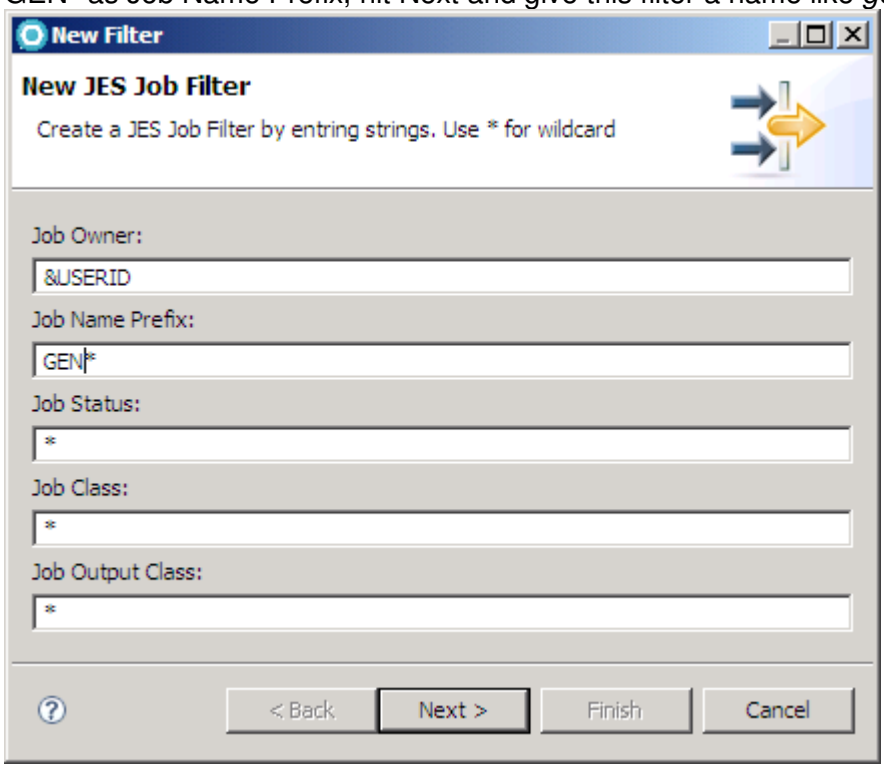
62. Expand My Data Sets clicking on the + and you should now have all the required datasets for the next exercise ready.

JES job filters are used to define the search that is done in JES for jobs. JES job filter search parameters include the following: Job Owner, Job Prefix, Job Output Class, Job Status and Job Class, comparable to SDSF. You can use * as a wildcard character.

The filter “My Jobs” you just used has been created for you by default. To create a new filter (you don’t have to do this now), right-click the JES node and select New → New JES Job Filter…



You can define several criteria to limit the jobs displayed. For a later exercise use the prefix GEN* as Job Name Prefix, hit Next and give this filter a name like *generated*.



1.5 Explore USS

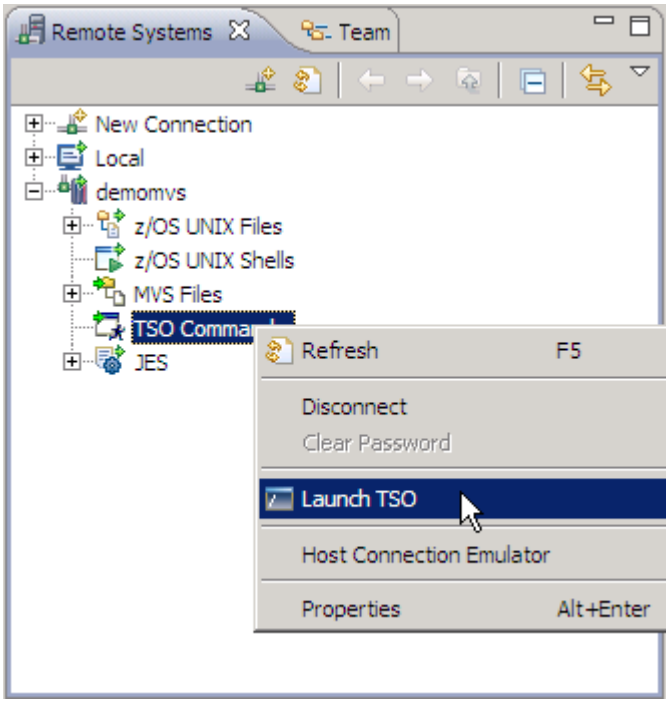
63. To display the built-in USS File Filters, expand USS Files. The three built-in USS filters are My Home, Home, and Root. Expand My Home to see the contents of your home directory.

Note: The My Home filter displays your home directory. The Home filter displays the directory containing your home directory. For most systems this will be '/u', but this may vary from system to system. The Root filter displays file starting with the file system root (i.e. '/'). If you want additional filters, you could right-click 'USS Files' and from the context menu, select Add Filter. Note that like the other filters, you can specify multiple criteria by right-clicking the filter, and from the context menu, selecting Change or Properties…In future lab exercises, we will be working with your home directory. Since there is a built-in filter to display the contents of your home directory, we will not need to add any USS File Filters for the upcoming lab exercises.

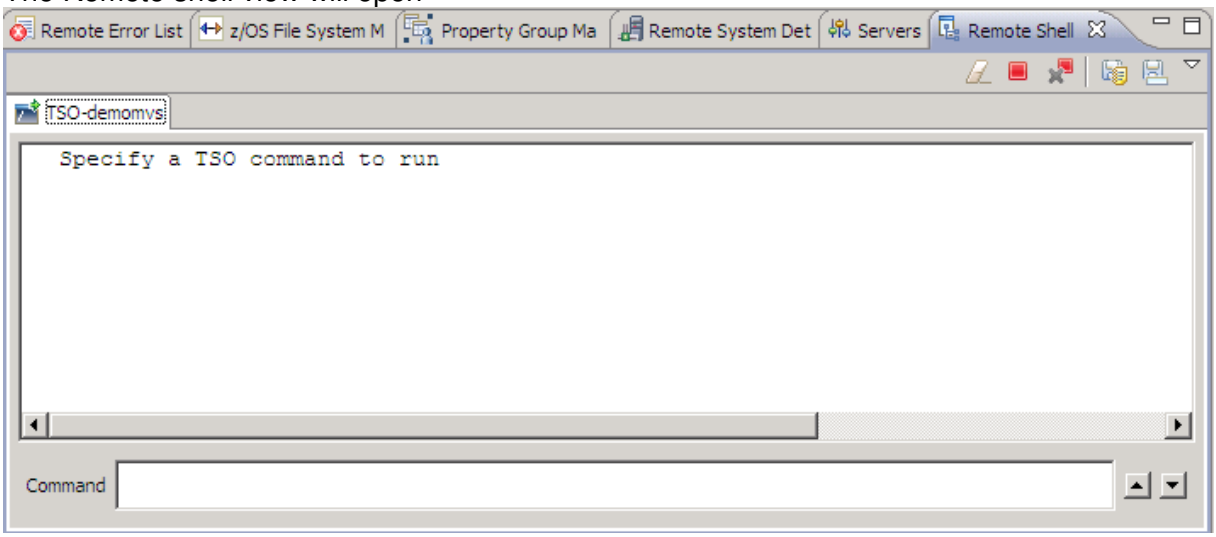
1.6 Optional - Explore TSO Commands

This is an optional exercise.
If you are interested, you can explore the TSO Commands feature of RDz
Users can launch TSO session from Remote Systems view, they could have multiple TSO Sessions up at the same time. The New TSO Commands UI is based on USS Shells by RSE. Let's see one example below.
Using the z/OS projects view and Remote System perspective, be sure that you are connect to the z/OS. If not connected, right-click on demomvs and select connect.

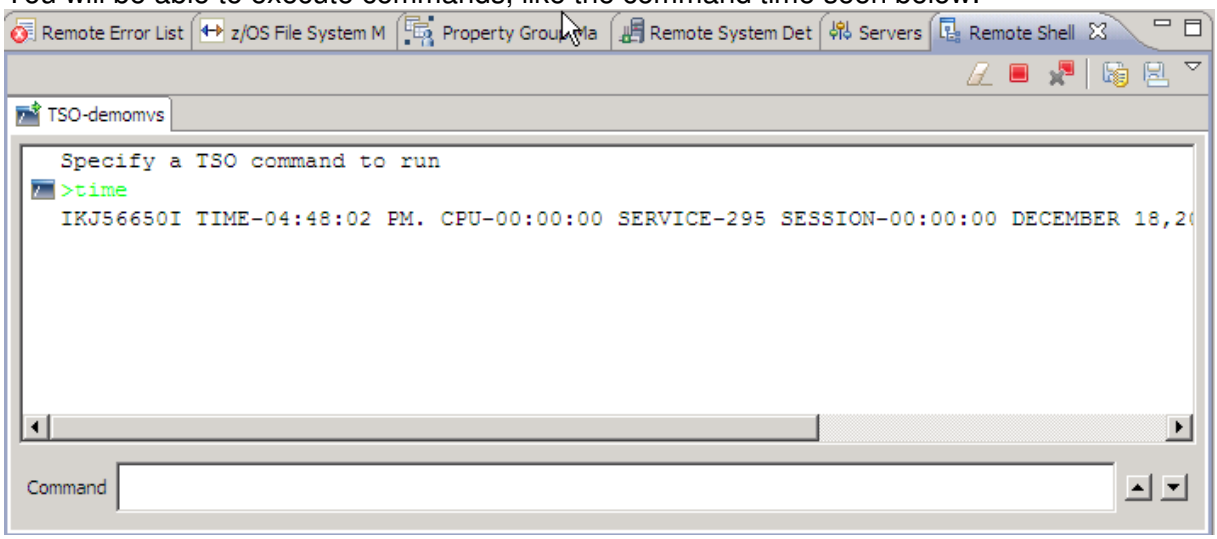
64. Scroll down to see the TSO Commands note. Right Click on it and select → Launch TSO as seen below:



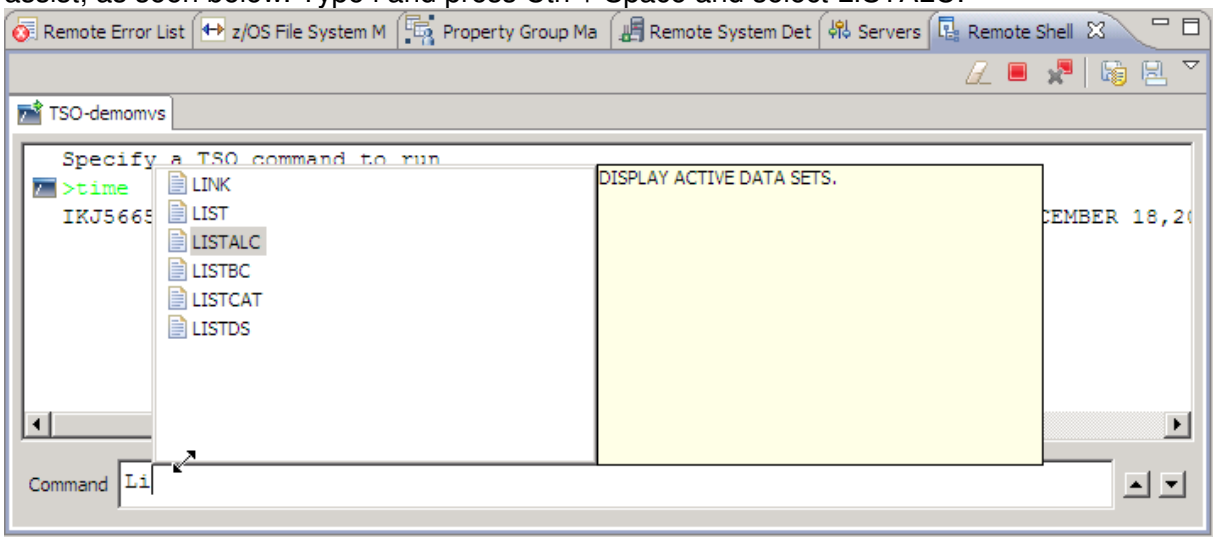
65. The Remote shell view will open



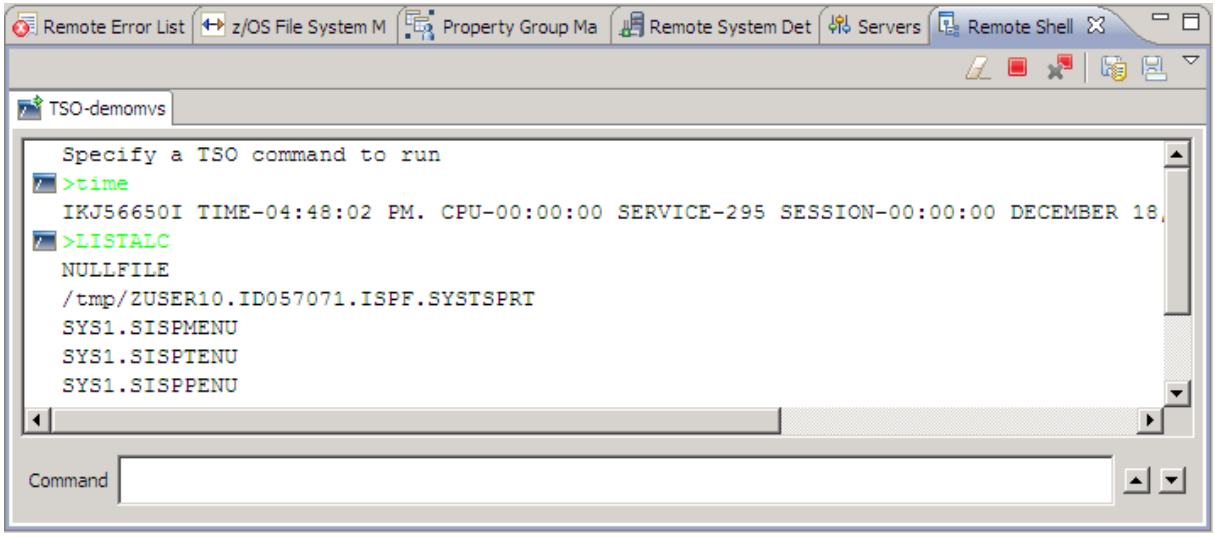
66. You will be able to execute commands, like the command time seen below:



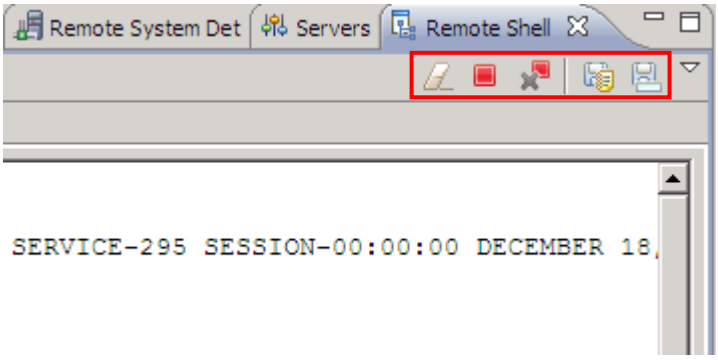
67. Also in the command line you will be able to use the Ctrl + Space to have the content assist, as seen below. Type l and press Ctrl + Space and select LISTALC:



68. The result will the one below



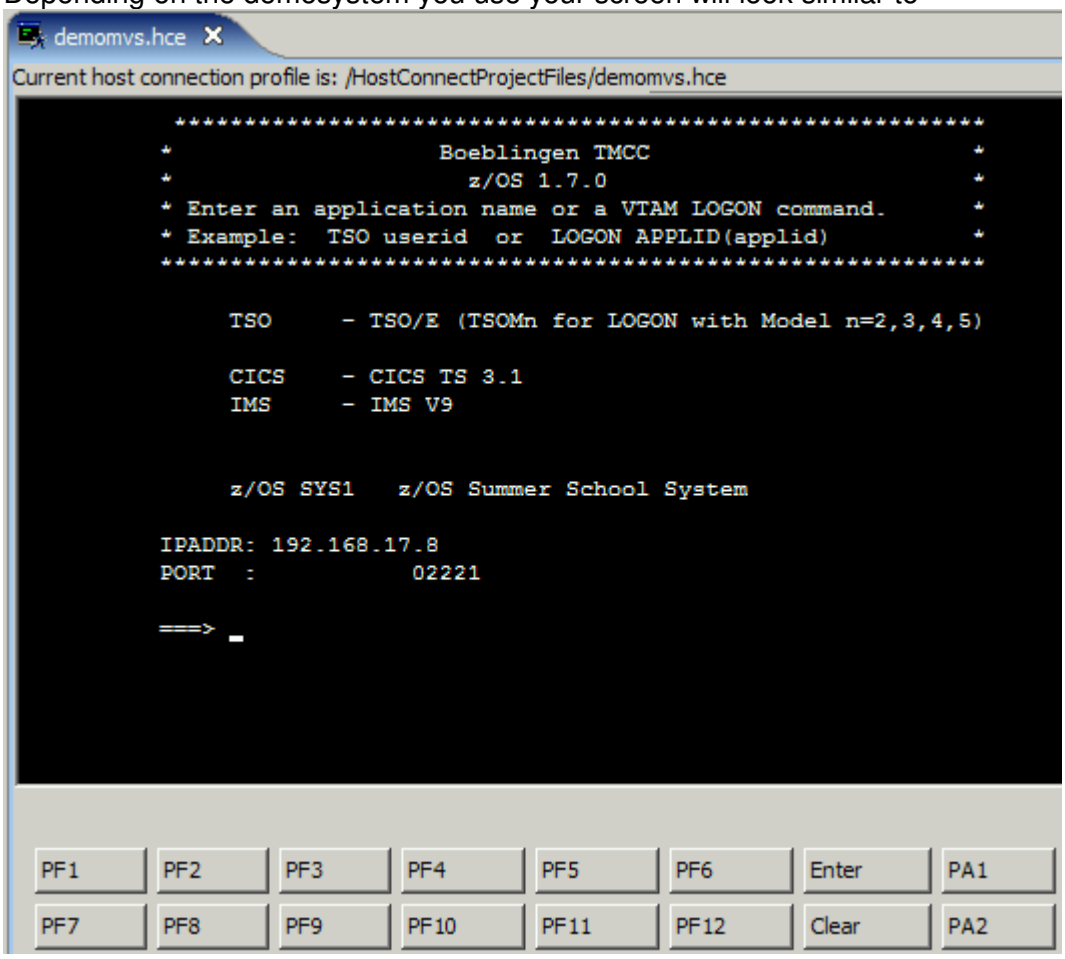
69. You also can use the icons below and perform some activities..



70. Right click on your remote project or anywhere within your demomvs remote connection. Select Host Connection Emulator Support from the context menu. You will be connected to z/OS.

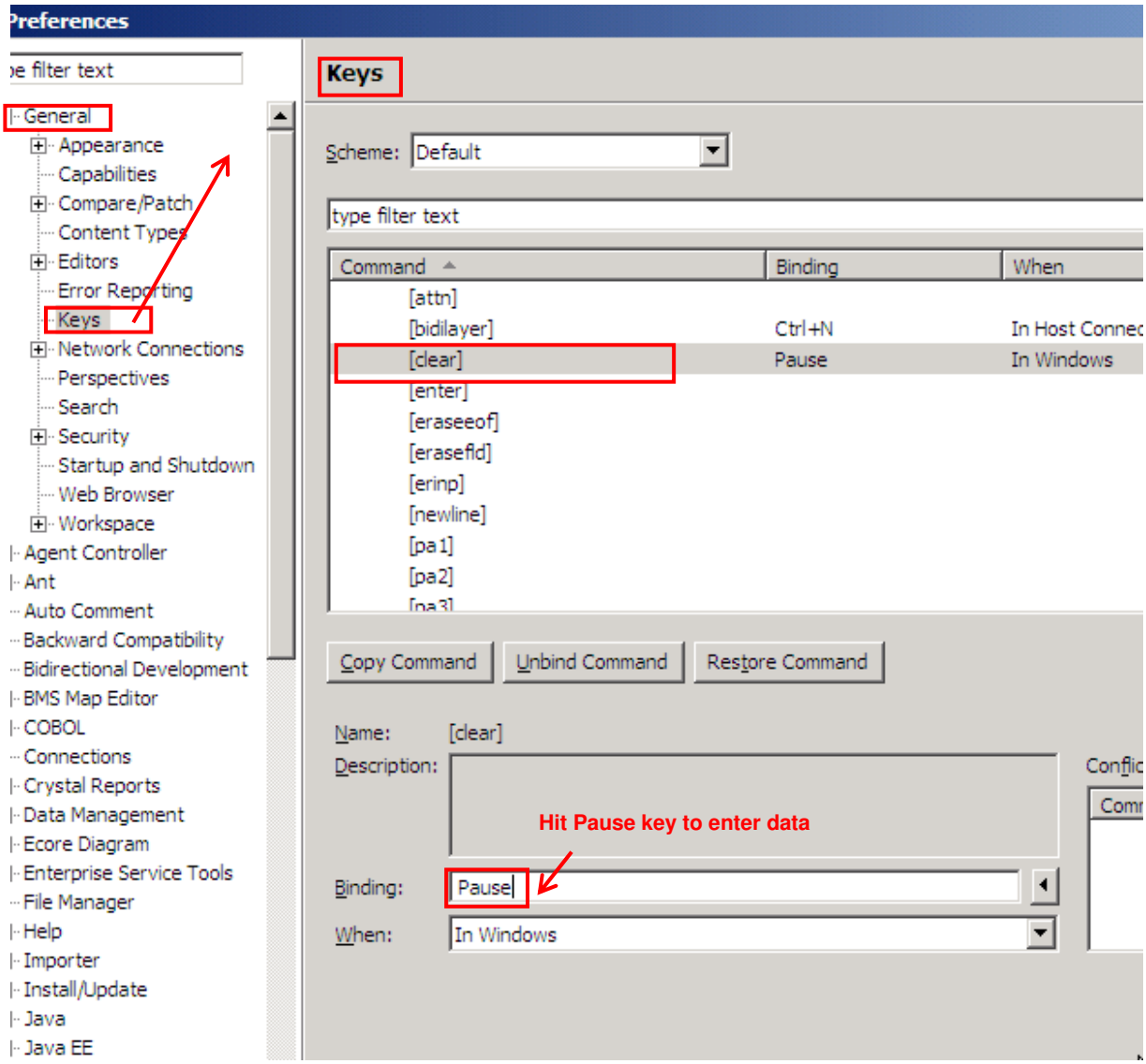
71. Resize the window so you will be able to better see the 3270 black screen. Just Double-click on the blue title demomvs.hce

72. Depending on the demosystem you use your screen will look similar to



73. Note that all PF keys are represented by buttons. Of course you can also use your keyboard. The default keys in addition to the PF keys are:
[Clear] – Esc
[Enter] – Return or Ctrl key
You can change those keys via Window → Preferences (from the menu bar) and then expanding General and selecting Keys.

74. For example to assign the Pause key to the [Clear] command as used in IBM Personal Communications select the [clear]-command, put the cursor to the Binding field, then hit Pause and at last hit Apply (which is not shown in the picture).



75. Click OK to save your changes and exit.

76. You can now optionally follow the screen instructions to logon to TSO (usually done by typing “TSO” and hitting [Enter]). Use your z/OS assigned user id USER##. Do not forget to logoff properly before closing the emulator.

2 Work with remote files

Working in eclipse is traditionally realized with projects. For z/OS this means that you will link your assets into a project structure: into z/OS projects. These projects can have USS or MVS subprojects, depending on whether you want to work with files residing in USS or members in MVS. In both cases your assets stay on the host and are only logically linked into sub projects.
By associating Property Groups with those sub projects you can bring your members or files into a context with compile, link and execution properties.

This chapter will guide you through the process of creating a remote project and setting its properties for different languages and runtimes.

2.1 Retrieve and import property group

Property Groups are a new feature of RDz 7.5. They contain the definitions for the z/OS Projects to specify build properties.
In earlier versions, properties needed to be defined for each project (respectively subproject) separately. Now all properties are kept separately and managed in so-called Property Groups which are associated with a remote connection. They can easily be imported, exported, copied, modified and associated with projects or members.
We will import the Property Groups for our lab which are stored in a XML file on the host. We will drag and drop this file from our remote system (USS) into a local project and import it.

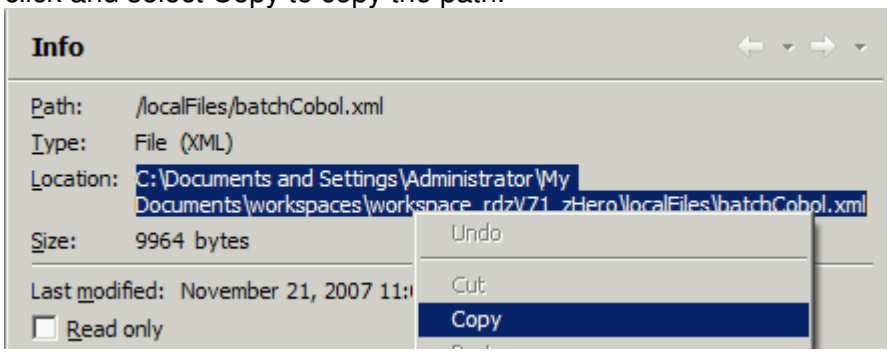
77. Open the Navigator View by selecting Window → Show View → Other… Expand General and choose Navigator. Move the Navigator View if you like it in another position.

78. Create a new general project by selecting File → New → Project and then General → Project. Name it "localFiles". This project is just a place to store our local files.

79. Go to the Remote Systems View and expand USS files and Root. Navigate to /local/zoaws/props (this is a directory on your demomvs, not local on your PC!)

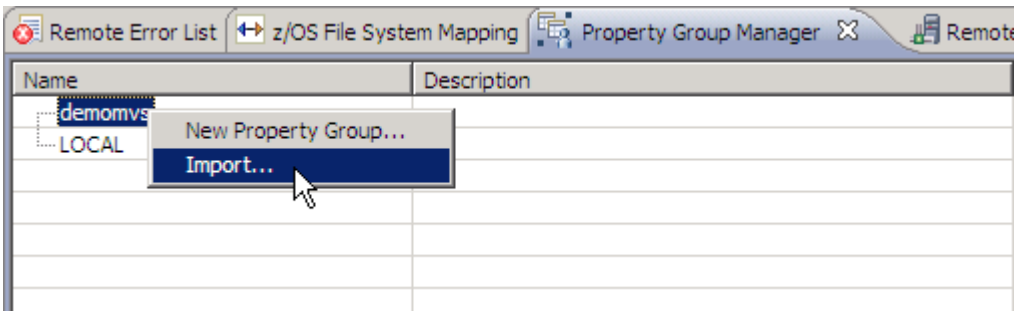
80. Select remoteProps75.xml and drag and drop it into your localFiles project in the Navigator View (Note: this will only work with the Navigator View).

81. To ease your search for this file in the file system later, we will copy the location. Right click your properties file and select Properties. Mark the path specified for Location, right click and select Copy to copy the path.



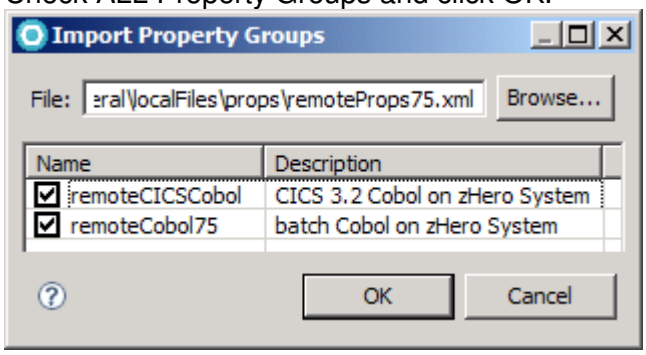
Now our properties file is available and ready to import.

82. To import the Property Groups, locate the Property Group Manager View at the lower panel group, rightclick on demomvs and select Import from the context menu.



83. Now either paste the content of your clipboard to the file destination field or use the browse option to locate the file.

84. Check ÁLL Property Groups and click OK.

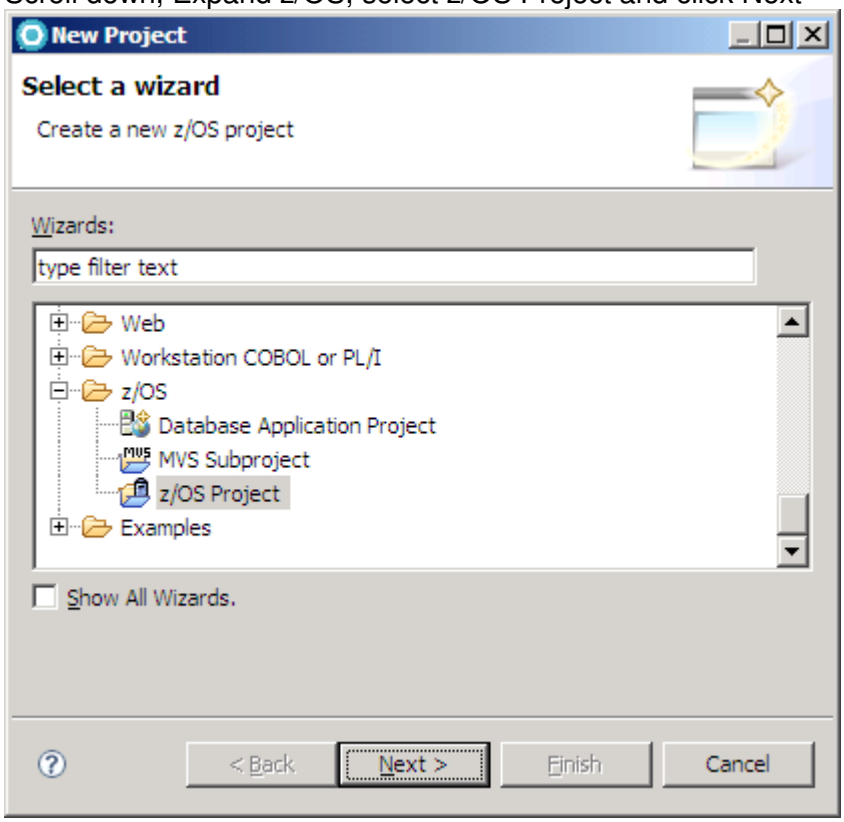


You now see the property group as child of the remote system connection.

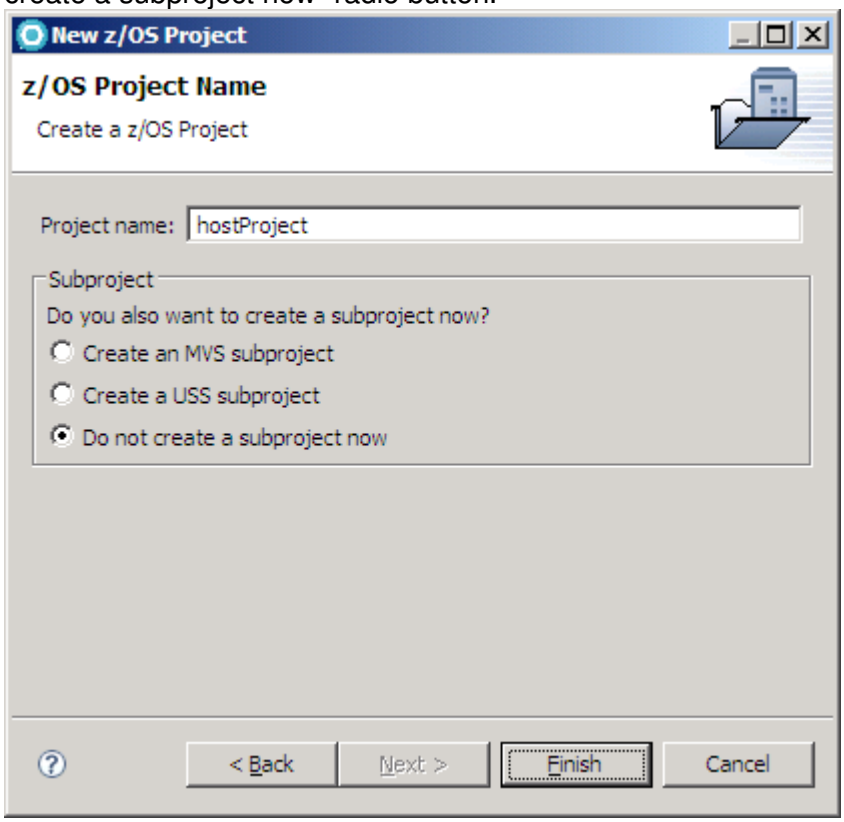
2.2 Create an remote project

85. Select File → New → Project… from the menu bar

86. Scroll down, Expand z/OS, select z/OS Project and click Next



87. On the New z/OS Project panel, name the project “hostProject” and select the “Do not create a subproject now” radio button.



88. Click Finish. This will create a z/OS parent project.

We will now create an MVS sub project.

89. You can define the MVS project only when you are connected to the system. Make sure you are connected in your Remote Systems view.

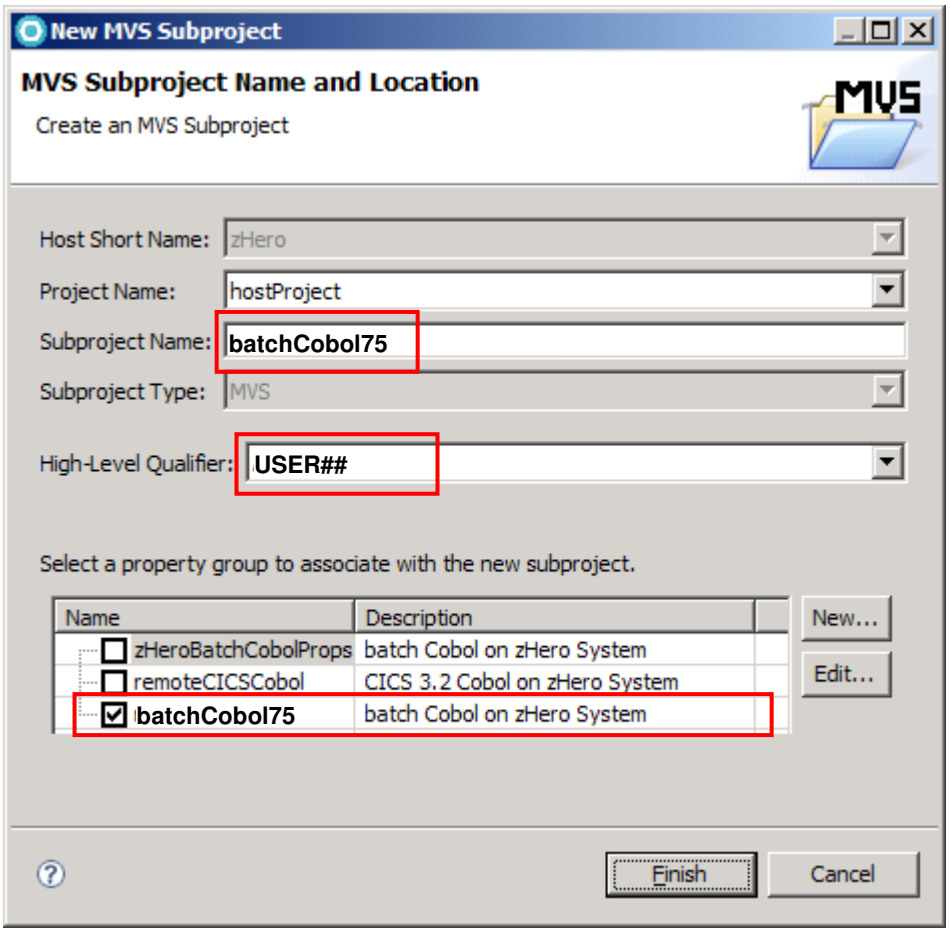
90. Go to the z/OS Projects View, right-click hostProject that you just created and select New → MVS Subproject from the context menu.

91. Depending on which lab you are going to work on check the Property Group with the name in the following table:

	Batch	CICS	IMS
COBOL	batchCobol75	CICSCobol75	
PLI			
C/ C++			
Assembler			

Table 1 - available Property Groups for different languages and runtimes

92. Name your MVS sub project like the name of the Property Group (the screenshot shows an example) and select your userid as high level qualifier. Click Finish.



93. You should see an MVS Project in your z/OS Projects view. If you click on the + sign to expand the project, you'll see that is empty.

We will review the imported properties later.
This lab will continue with batch COBOL programming to demonstrate more RDz features and functions. Please look at Table 1 - available Property Groups for different languages and runtimes - to see which other languages and runtimes are currently available for this workshop. Please contact your lab administrator to receive other lab material.

3 COBOL

In this chapter you will learn how to work with remote Cobol in batch. You already defined an MVS project and imported the property group – now we will have a more detailed look at the Property Group itself.

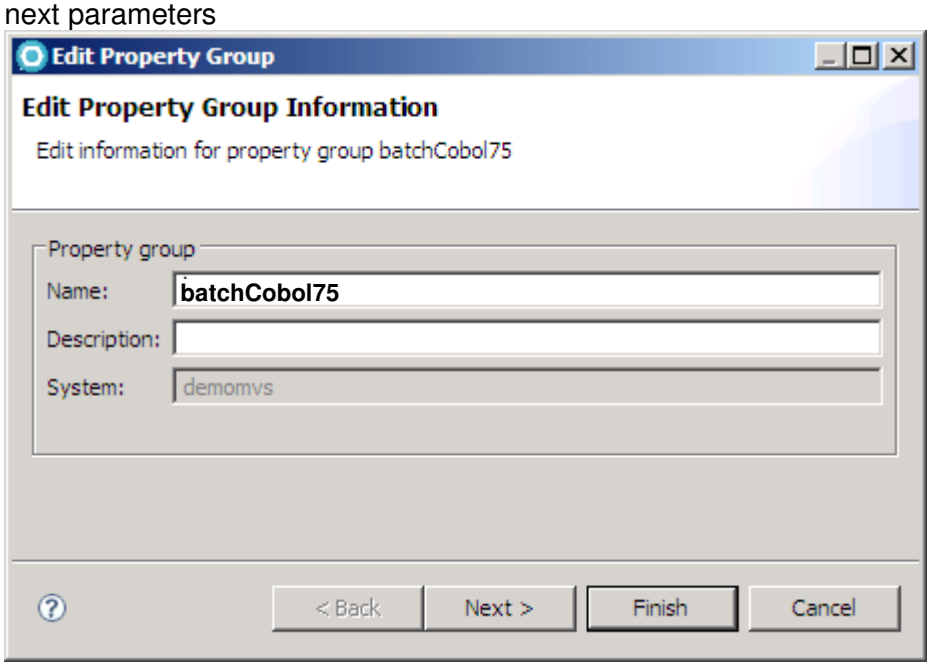
You will start with a very simple HelloWorld program (mostly generated) that you will edit, compile, link and run. Later on you can have a look at some more complex programs for debugging and other features like dependency check.

3.1 Verify project properties

Before you start to work with remote COBOL programs let's have a look at the property group you already associated with your remote MVS subproject.

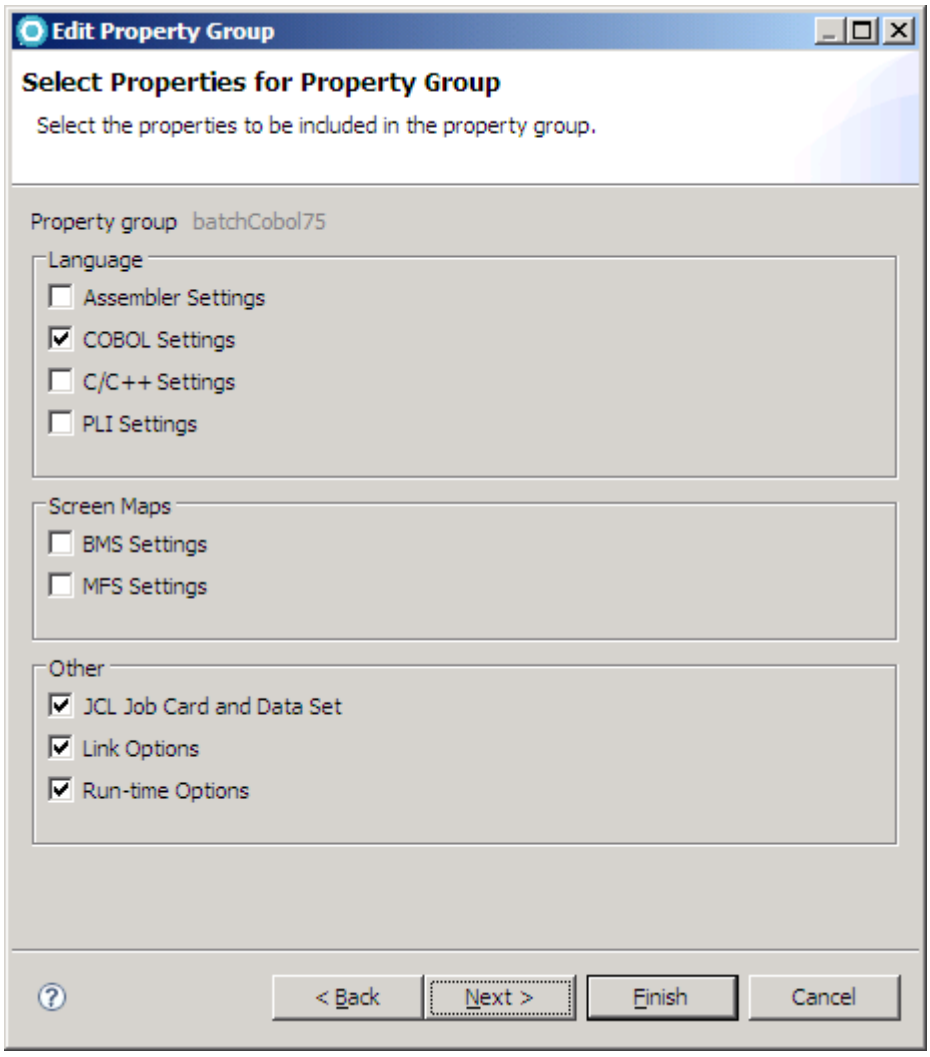
94. In the Property Manager Group, expand the remote System demomvs, then Right-click your batchCobol75 property group and select Edit.

95. At first a short overview about the title and a description is given, click next to go to the next parameters

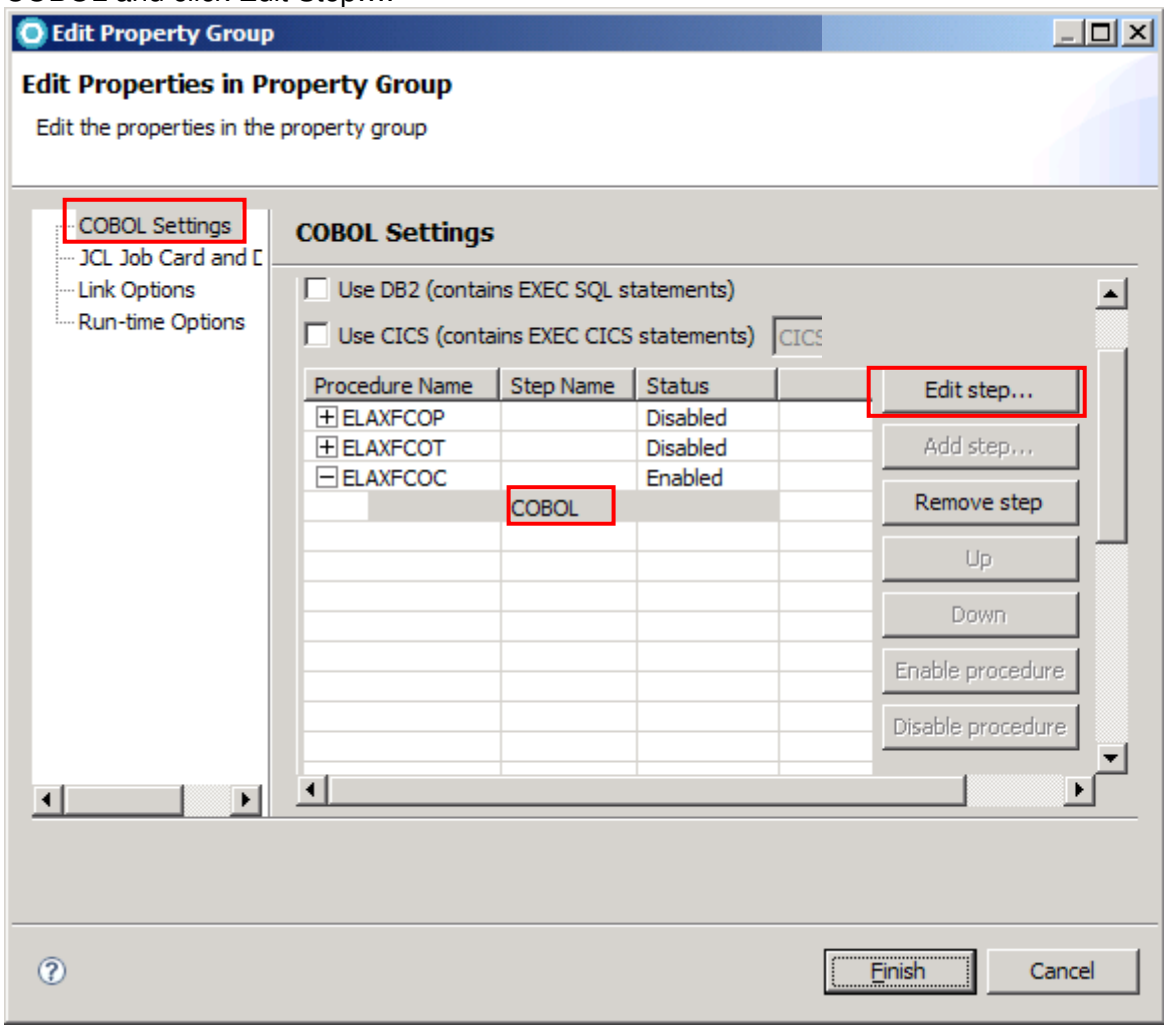


:

96. The next panel offers you to select the property you want to set. This is one of the major improvements of the new version regarding the project property groups. Please verify that the following options are ticked, then hit Next:

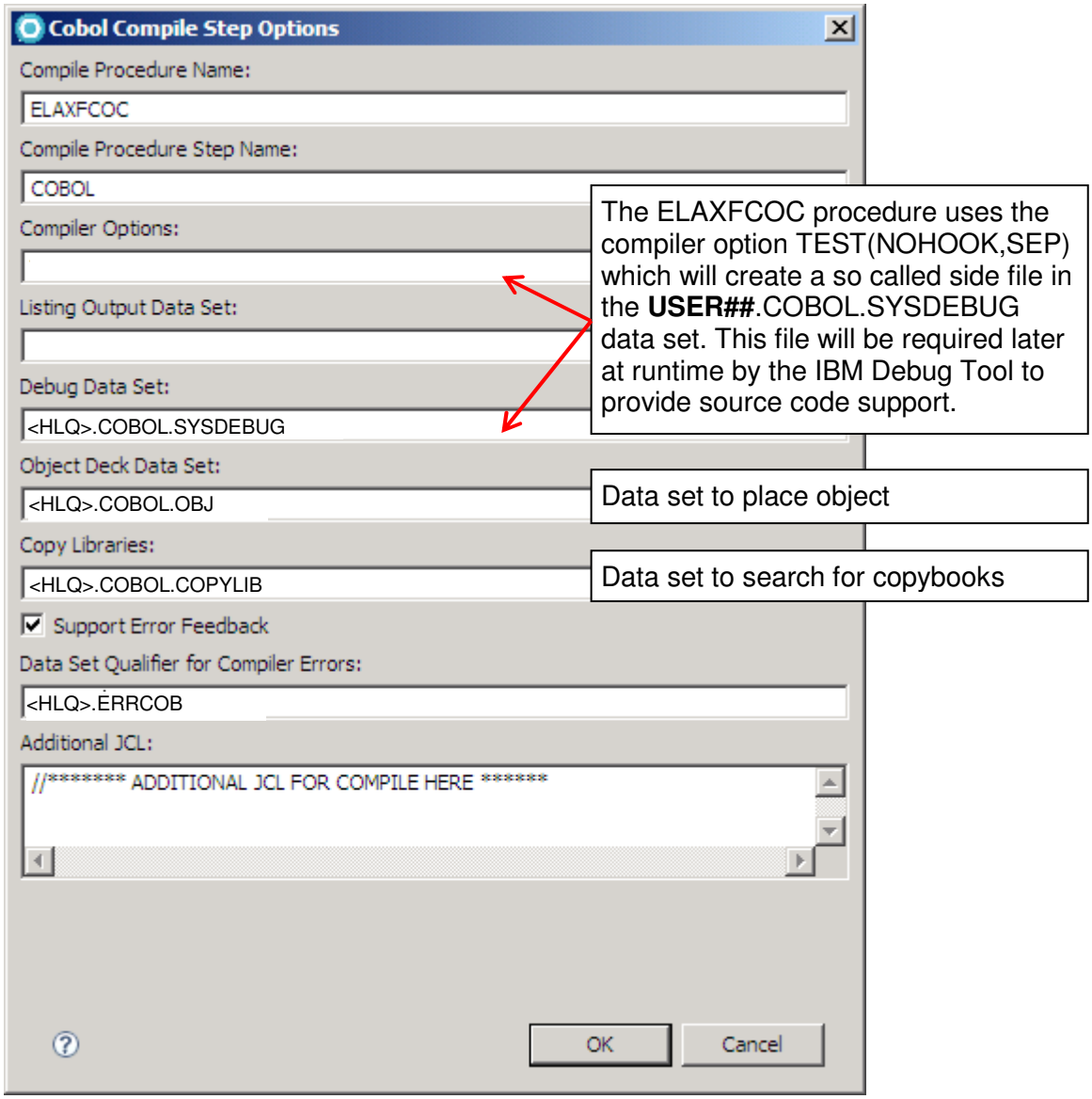


97. The COBOL Settings panel should be next, expand ELAXFCOC procedure, select COBOL and click Edit Step….

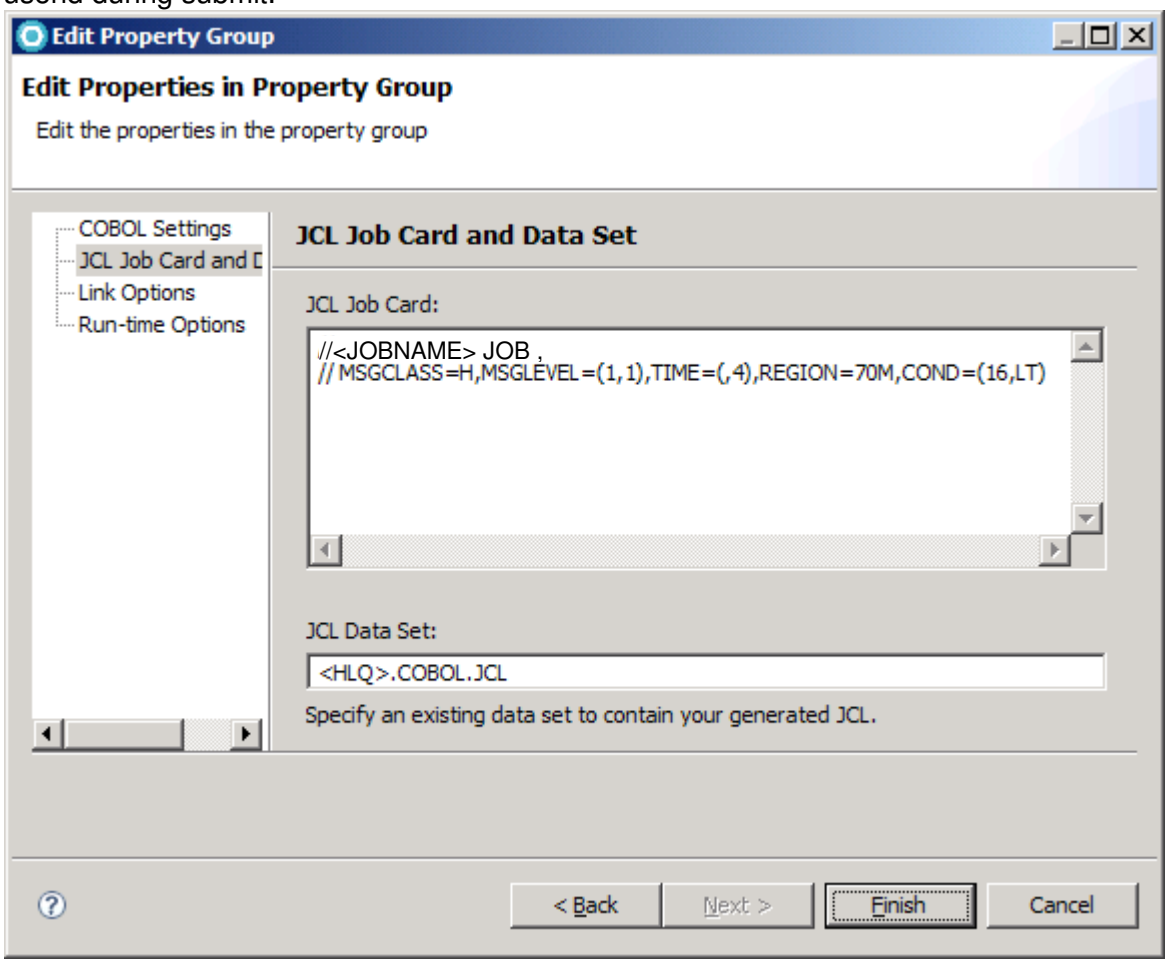


Note that we could enable and disable the procedures that must be executed before the COBOL compilation done by ELAXFCOC if necessary. The default supplied procedures (but disabled) are:
ELAXFCOP – invoke separate DB2 pre-compiler
ELAXFCOT – invoke separate CICS translator
But for this lab: do NOT invoke them.
By setting the checkmarks next to “Use DB2” or “Use CICS” you can invoke the integrated translator. Also, do not check them because we are working in Batch.

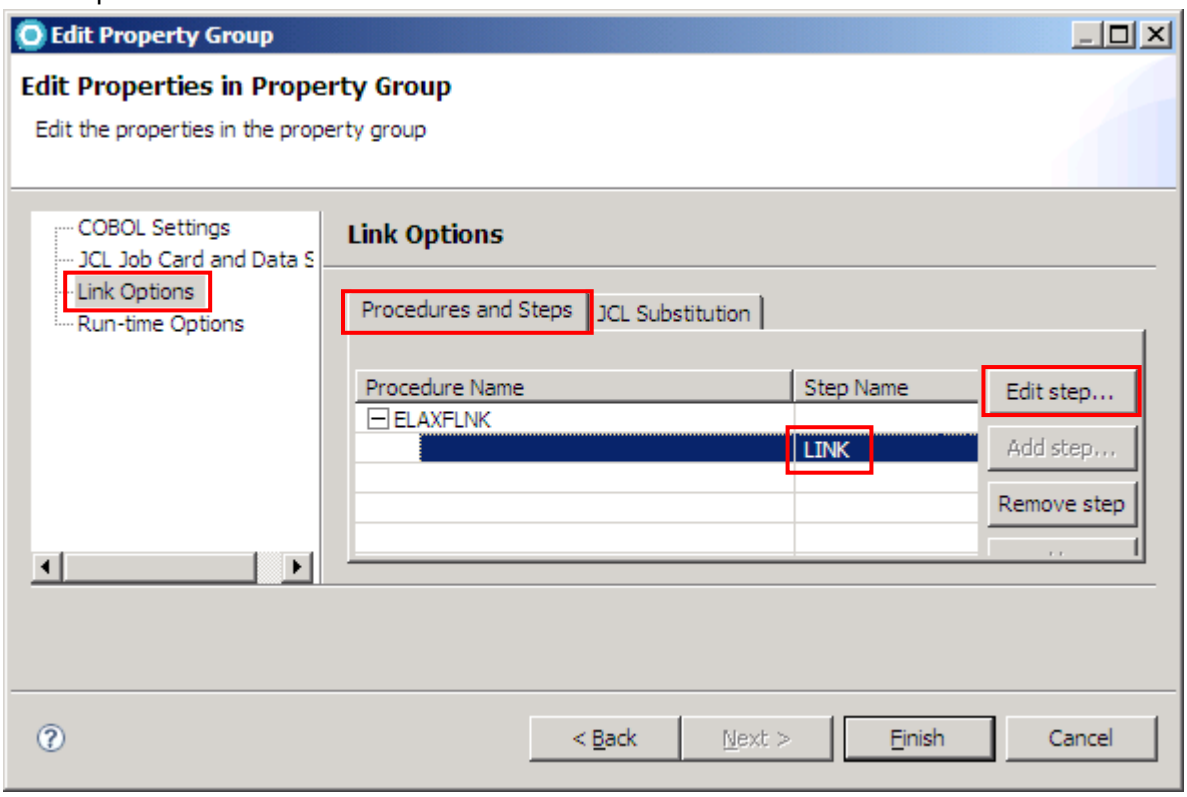
98. The COBOL Compile Step Options dialog will open. Review the settings. <HLQ> will be replaced by the userid you specified during creation of your MVS project. Select OK to confirm your changes.



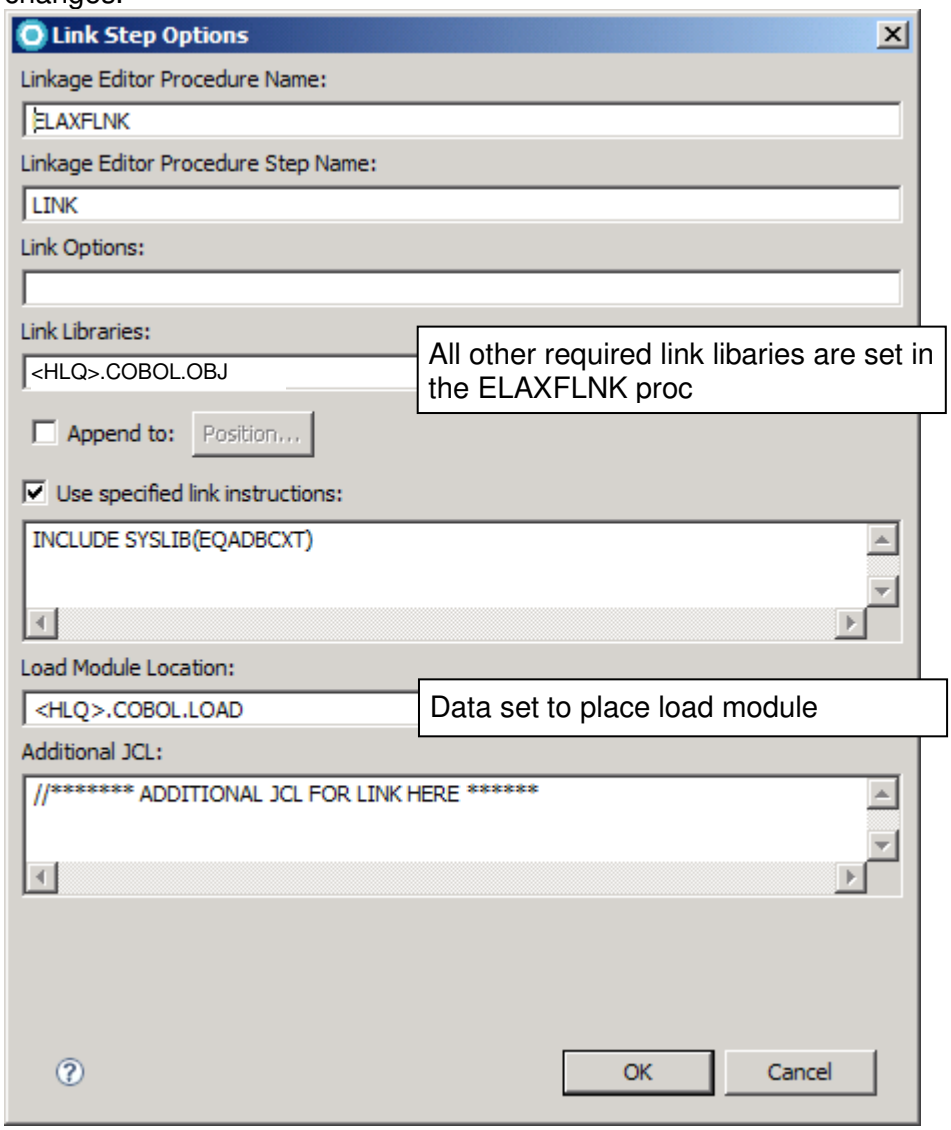
99. Select JCL Job Card and Data Set. <JOBNAME> will automatically be replaced by your userid during submit.



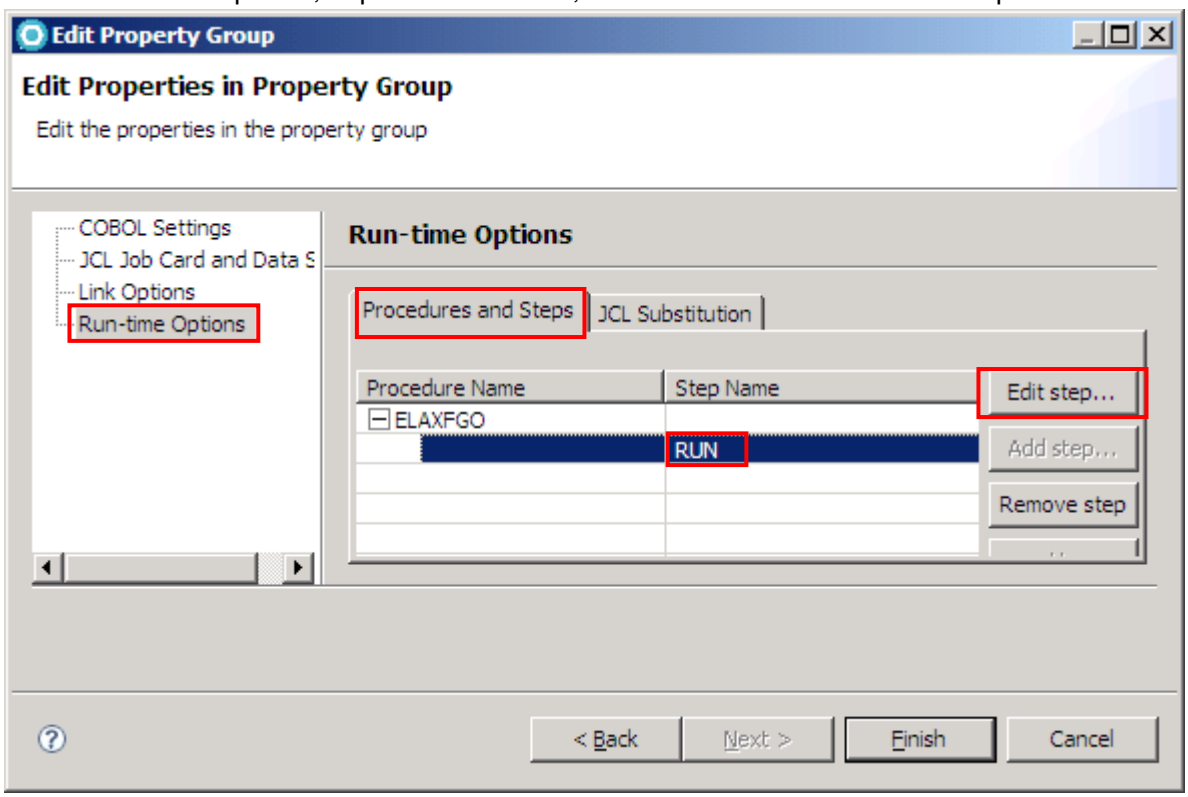
100. Select the Link Options. Expand the + besides ELAXFLNK, click on LINK and click on Edit Step…



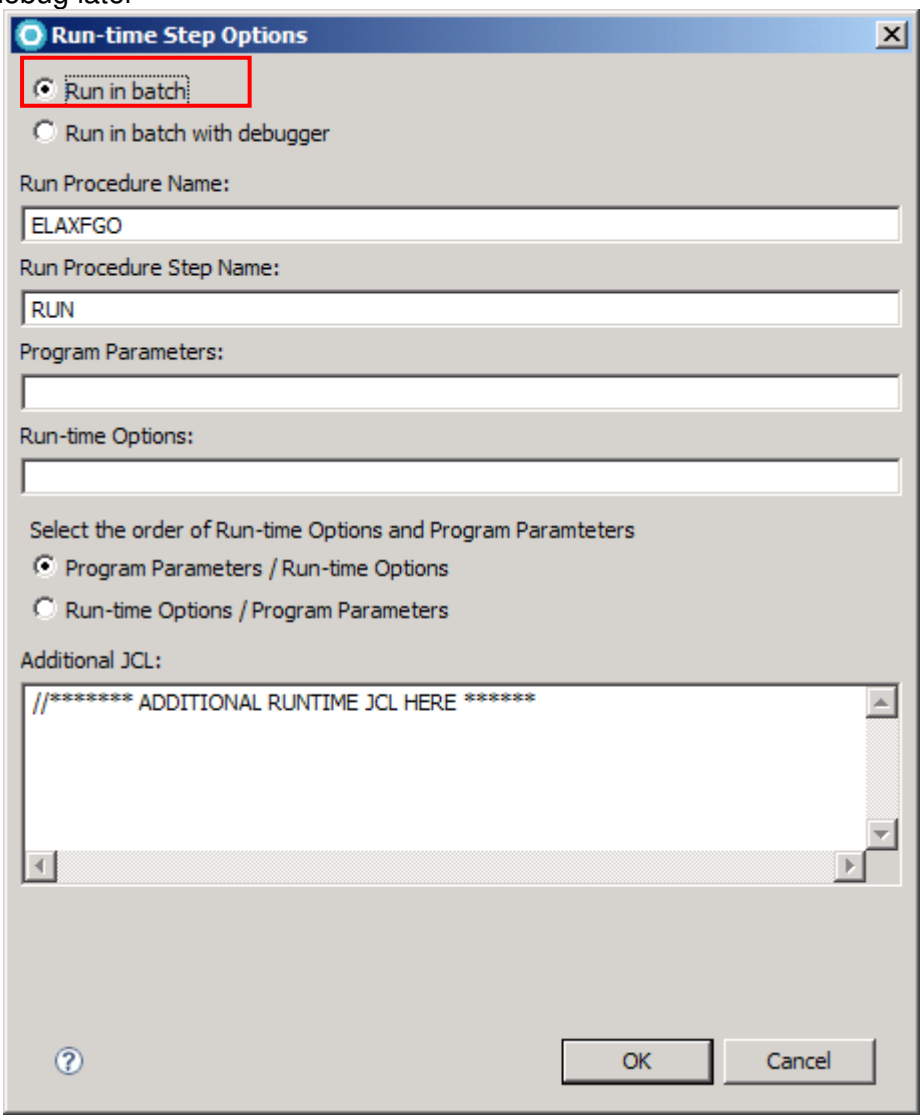
101. Adjust the link options to match the following picture and select OK to confirm your changes.



102. Go to Run-time Options, expand ELAXFGO, click on RUN and click “Edit step…”



103. Make sure that “Run in Batch” – not “Run in Batch with Debugger” is selected. We will debug later



.

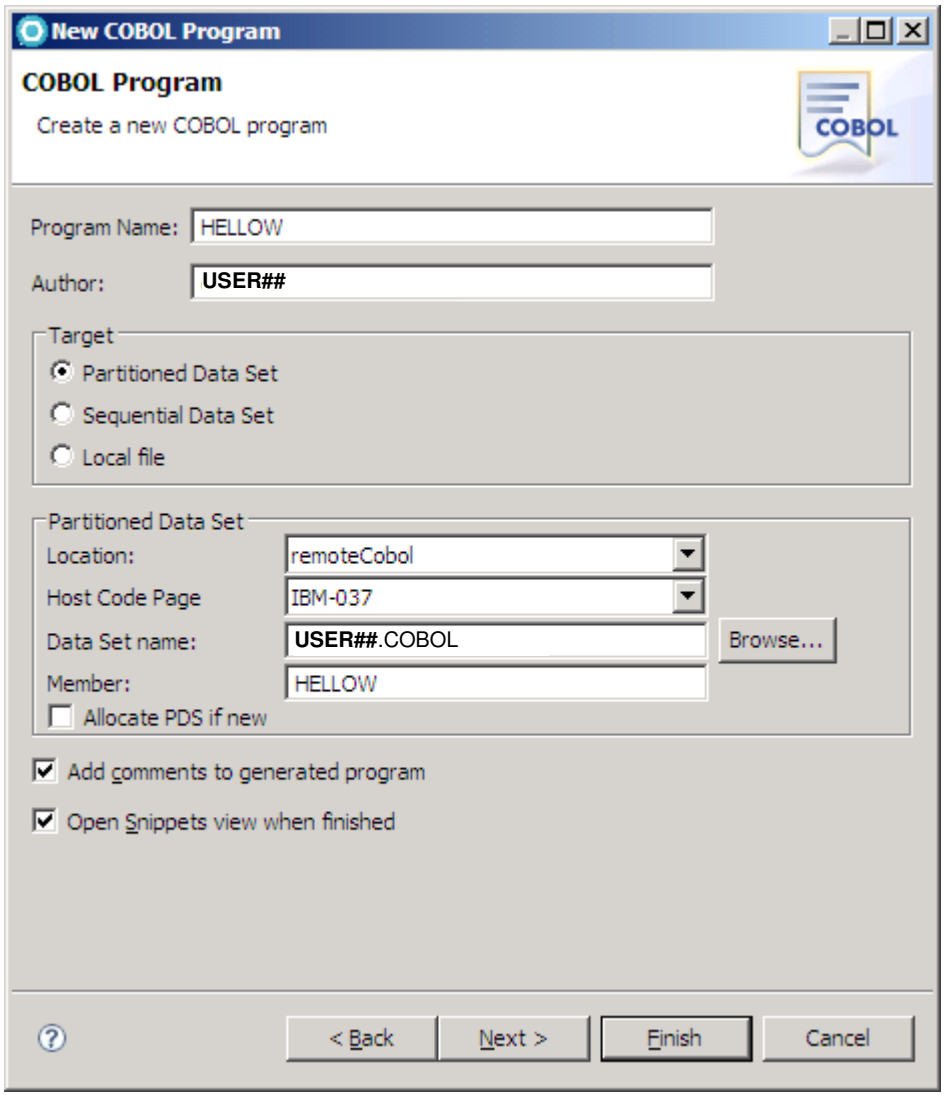
You have now seen all properties of the group that we are going to use in the next steps. To apply a property group we will associate it with a member, data set or project.

3.2 Create a COBOL HelloWorld program using zAPG

104. Select your COBOL dataset **USER##**.COBOL out of the z/OS Projects view.

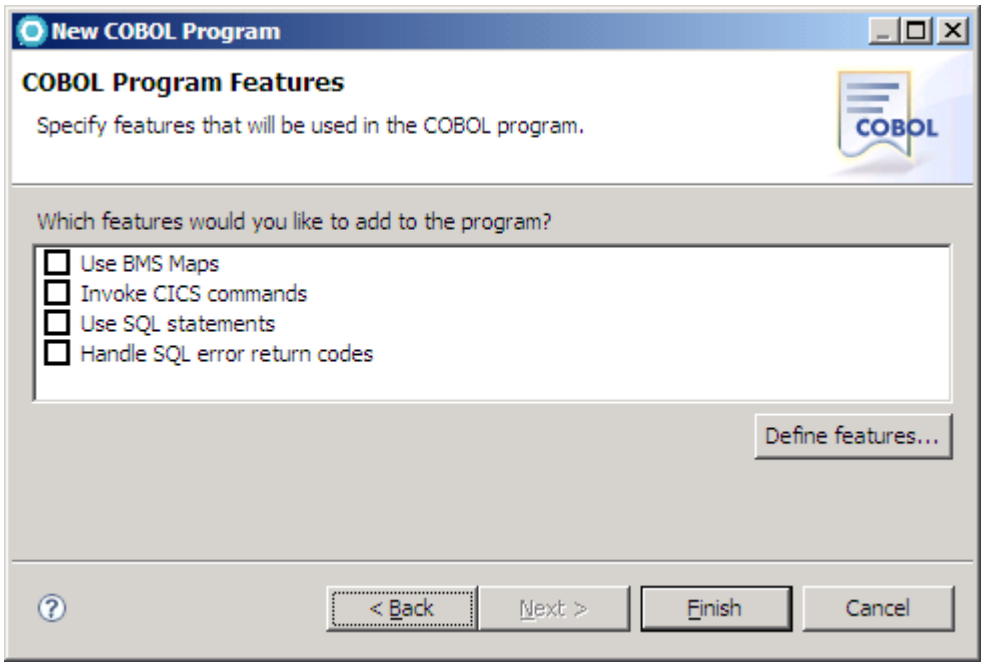
105. Select File → New → Other.. and choose COBOL → Cobol Program.
     Click Next.

106. Name your program and member HELLOW and place it in **USER##**.COBOL PDS.

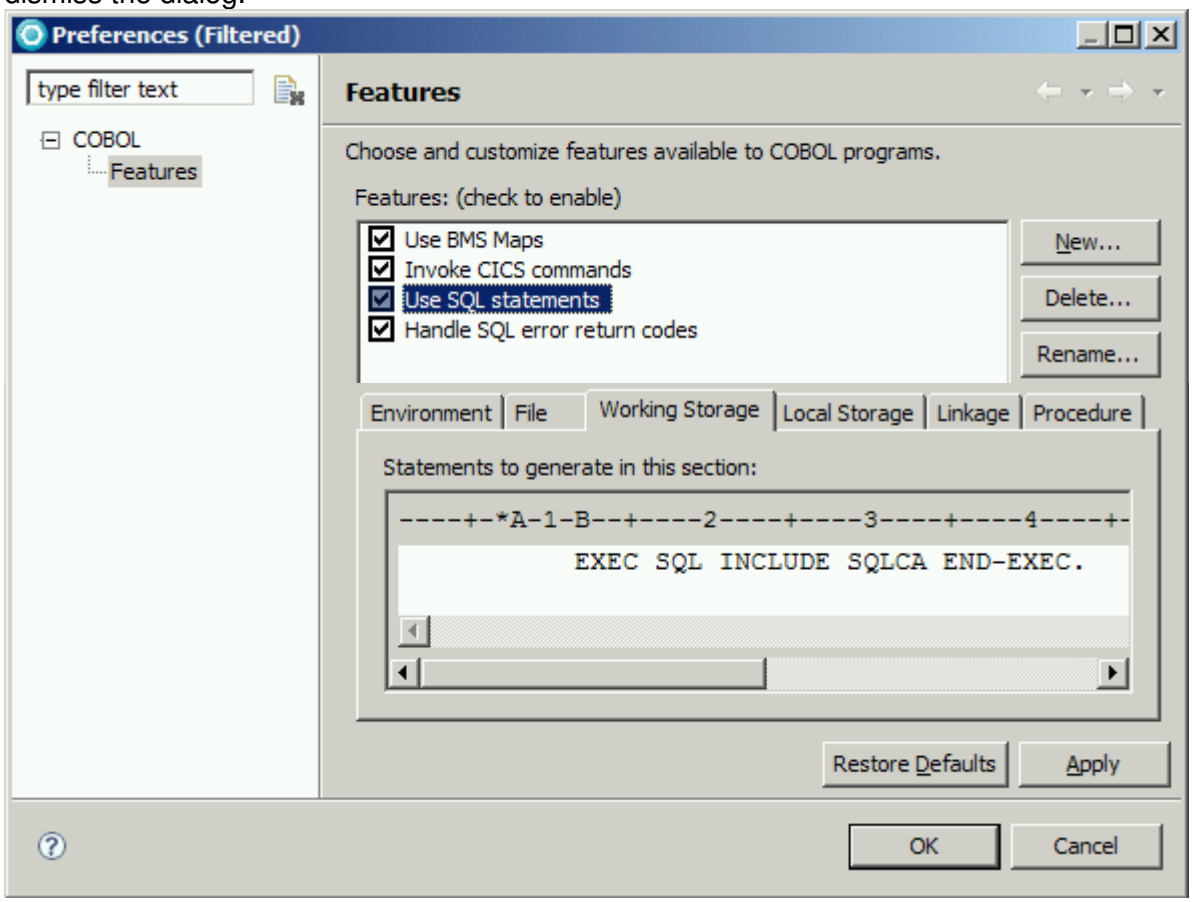


Click Next.

107. On the next panel you can specify features that will add code to your new program. Do not select any, but click on the “Define Features” button to have a look how a feature could look like.

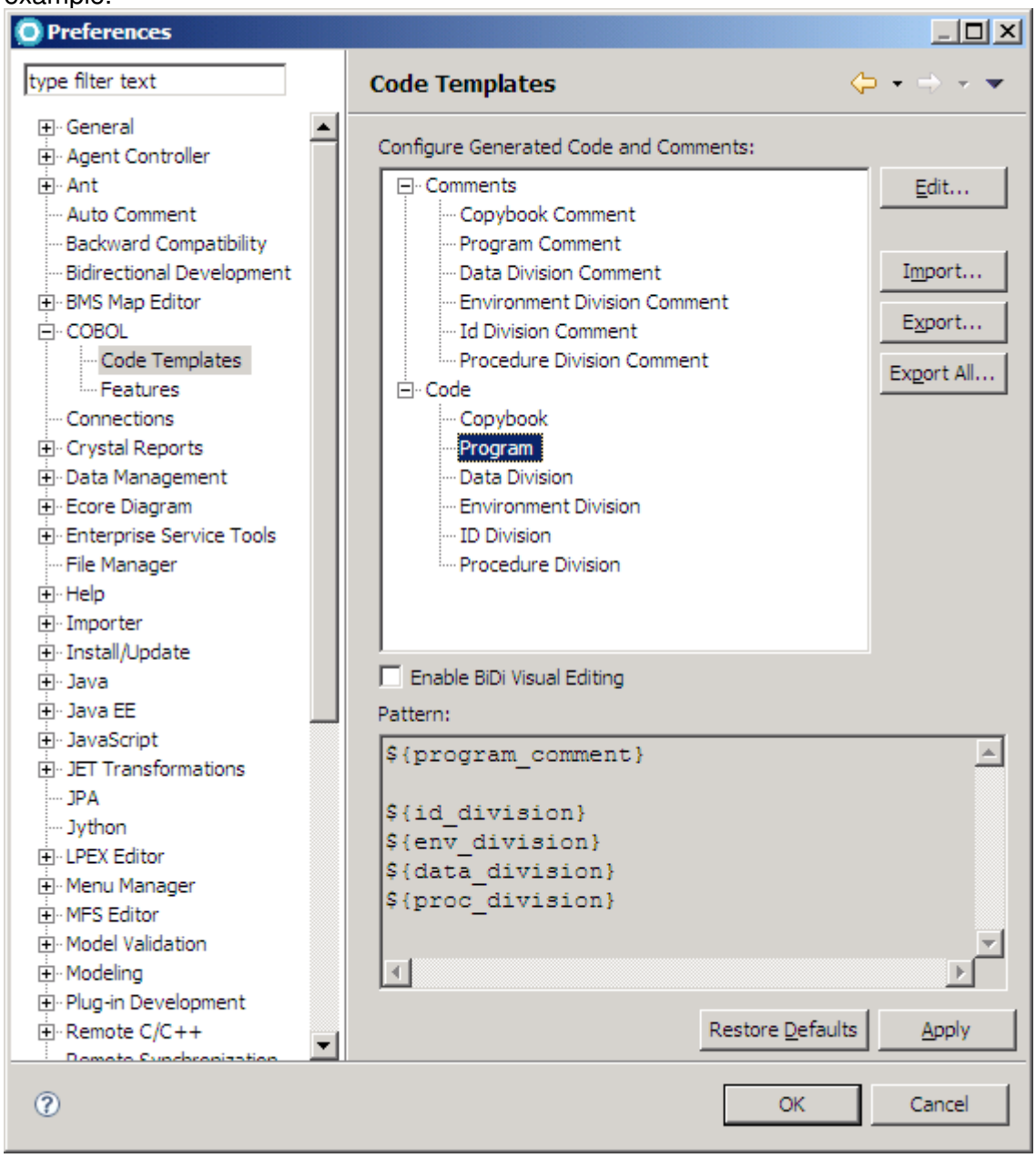


108. If you click on a feature the tabs below will show you what code will be added at which division or section. You could also create new or modify features here. Click Cancel to dismiss the dialog.



109. Click Finish and your program will be created. The editor opens automatically. A basic skeleton was created for you.

110. The preferences for this skeleton have been defined for your workspace. Click Window → Preferences and navigate to COBOL → Code Templates and you will see what led to creation of your program. You could use these templates to spread coding guidelines for example.



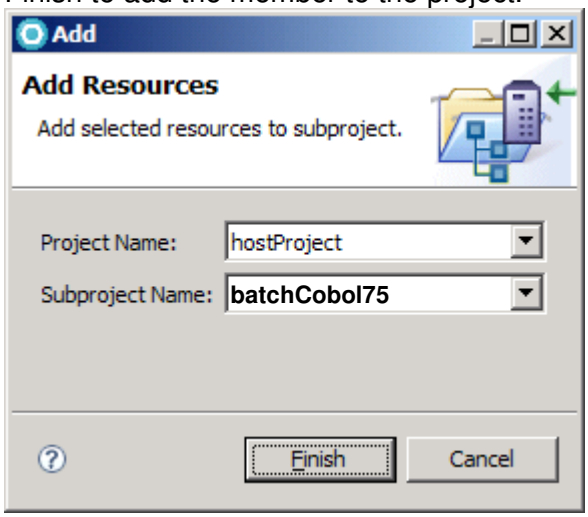
3.3 Add resources to your project

To make the new member available to your remote project, you will need to add it.

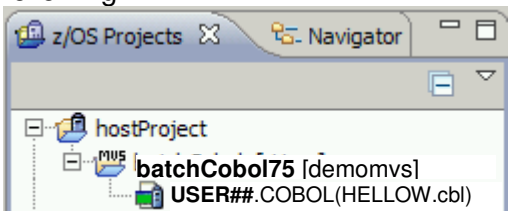
111. Locate your newly created member in your Remote Systems View and richt click it. (If you don't see your member, right click the filter or the corresponding data set and select Refresh).

112. Select "Add to Subproject…" from the context menu.

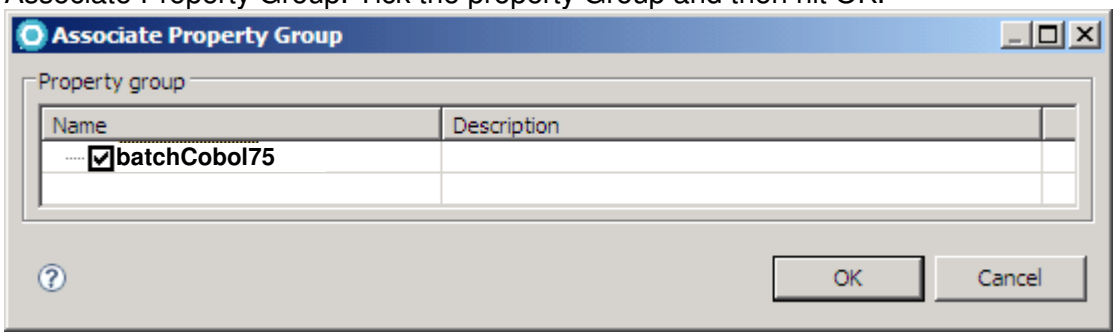
113. Accept parent project name hostProject and sub project name batchCobol75 and click Finish to add the member to the project.



114. Switch to the z/OS Projects view and you will see that USER##.COBOL has been added to the batchCOBOL project. The z/OS Projects view should look like the following:



115. Before we can proceed, we might need to apply a Property Group, which is new to version 7.5. Therefore right click the project batchCobol75 -> Property Group -> Associate Property Group. Tick the property Group and then hit OK.

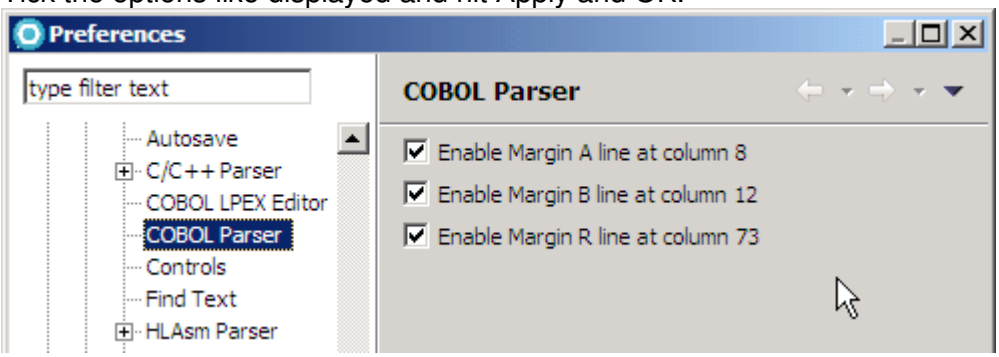


3.4 Exploring the Editor

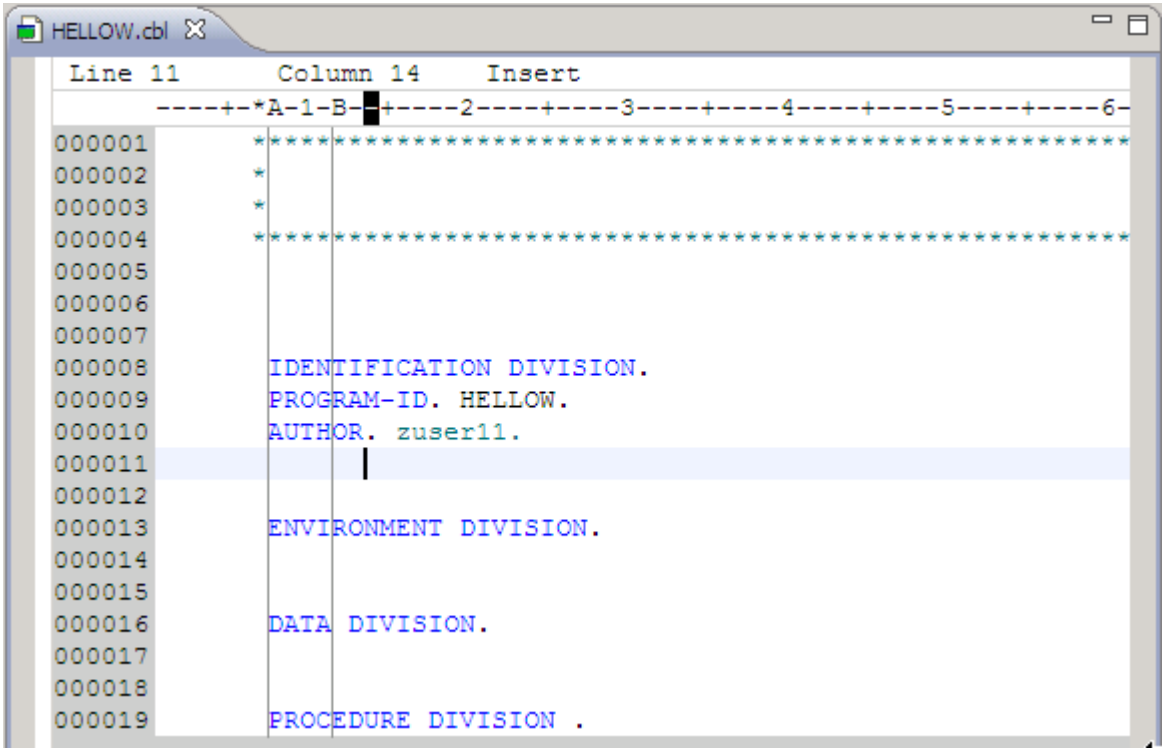
Code assist helps with the completion of code and is activated by pressing Ctrl + Space when in the content area of the z/OS LPEX editor.

116. In your HELLOW program navigate to the empty line below PROCEDURE DIVISION. Place your cursor in column 12 (B) (from beginning of the line hit TAB 2 times).

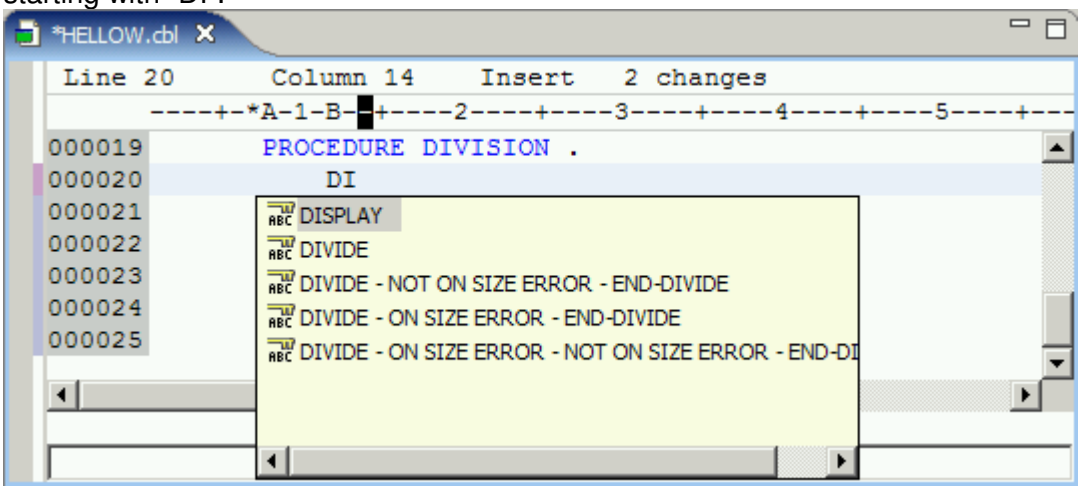
117. Due to the fact, that in Cobol the column in which the line of code starts is depending on strict rules, you might enable a margin line. You might enable this in the preferences → LPEX Editor → System z LPEX Editor → COBOL Parser
Tick the options like displayed and hit Apply and OK.



The result are the following margins:



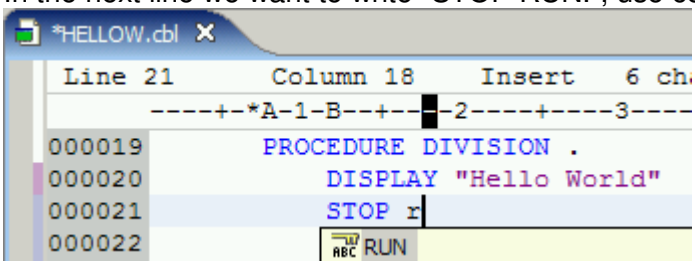
118. Type “DI” and press Ctrl+Space. A window will show all suitable Cobol commands starting with “DI”.



119. Press Enter to select the first one (DISPLAY) and insert the string “Hello World”.

120. Use Ctrl+ Enter to create a new line (like ISPF newLine command).

121. In the next line we want to write “STOP RUN.”, use content assist as well.



122. Explore the different code completion possibilities. Press the Esc key to dismiss the popup list. For COBOL verbs that require additional parameters (identifier in COBOL terminology), code assist will present data items defined in the working storage section as selections in the pop up list.

Finally your PROCEDURE DIVISION should look like

```
PROCEDURE DIVISION .
    DISPLAY "Hello World"
    STOP RUN.
```

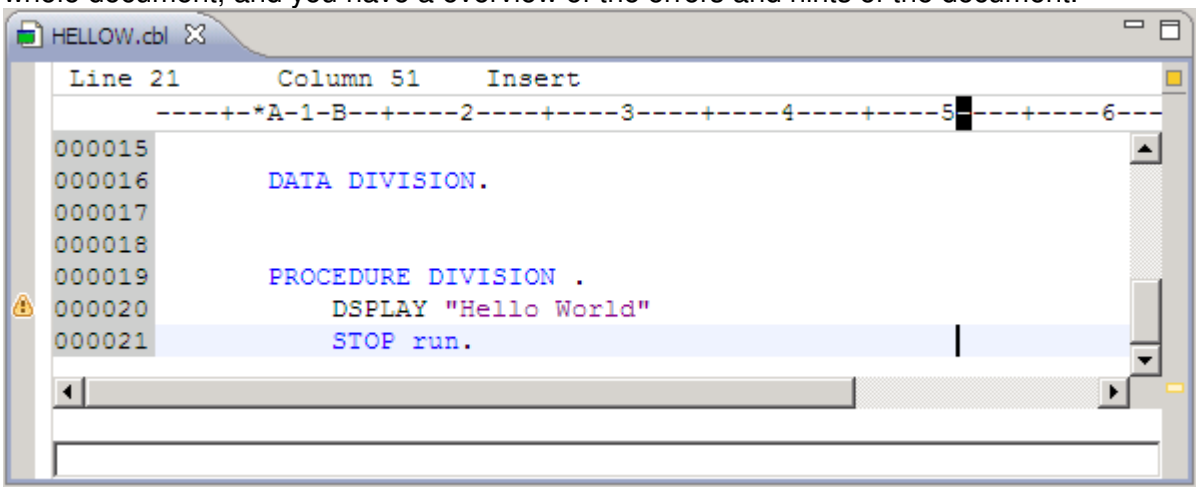
3.5 COBOL Syntax Checking

Before submitting a job to the z/OS system to compile the COBOL source file, you can perform a syntax check to ensure a clean compile.
You will deliberately introduce an error to illustrate the error feedback facility.

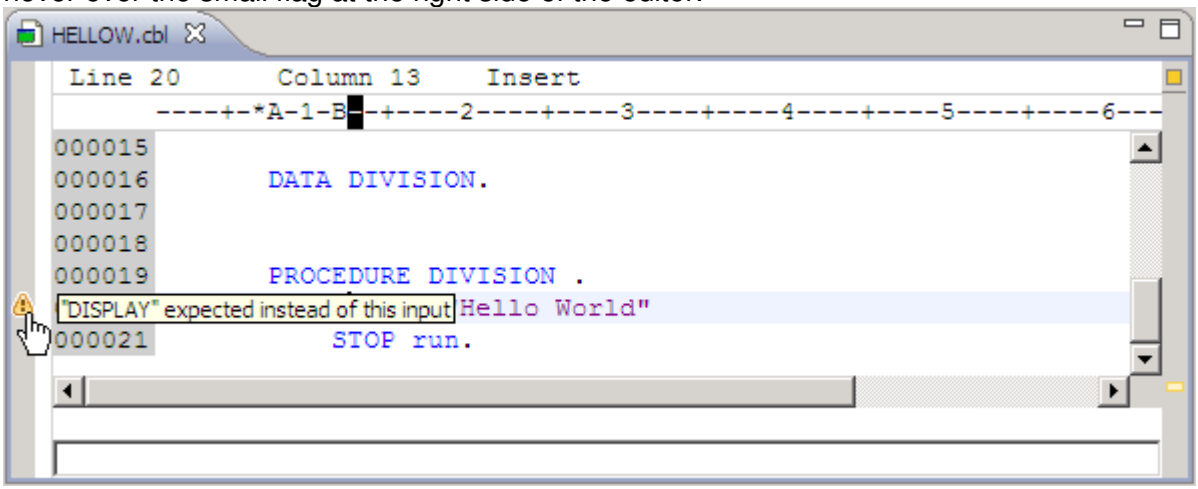
Since the new version 7.5 a new *real time* syntax check is implemented. When you do a typo a small exclamation mark is shown at the first column in front. This is inherited from the the distributed application development within eclipse.

123. Overtype DISPLAY with DSPLAY (NOT DISPLAY) to force an error and save your changes (CTRL + S).

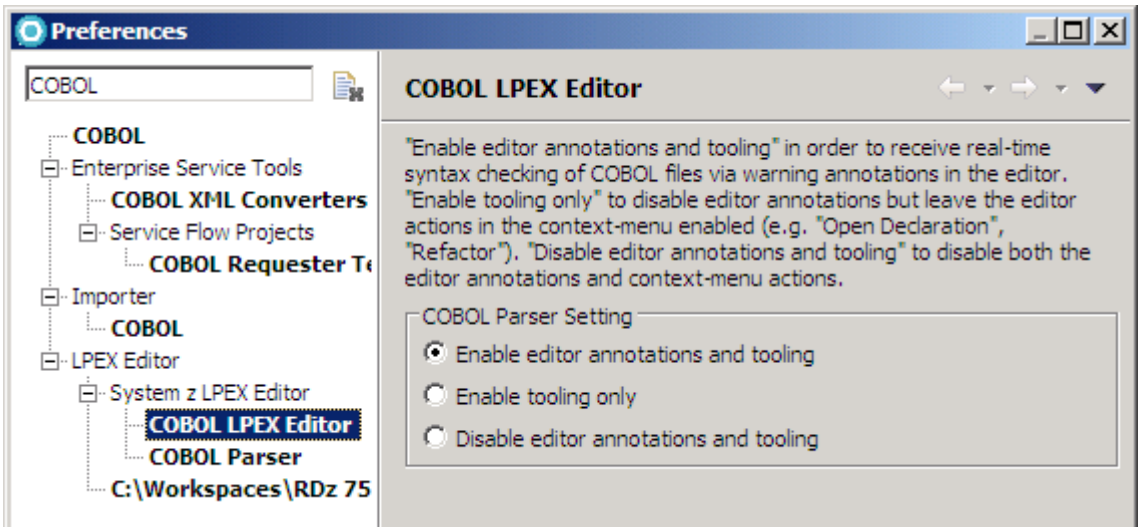
124. You now see already the yellow small exclamation mark in front of the line. At the right margin, please be aware of the small yellow-white flag. The margin represents the whole document, and you have a overview of the errors and hints of the document.



125. Now hover over the exclamation mark at the left – a hint is displayed. You can as well hover over the small flag at the right side of the editor.



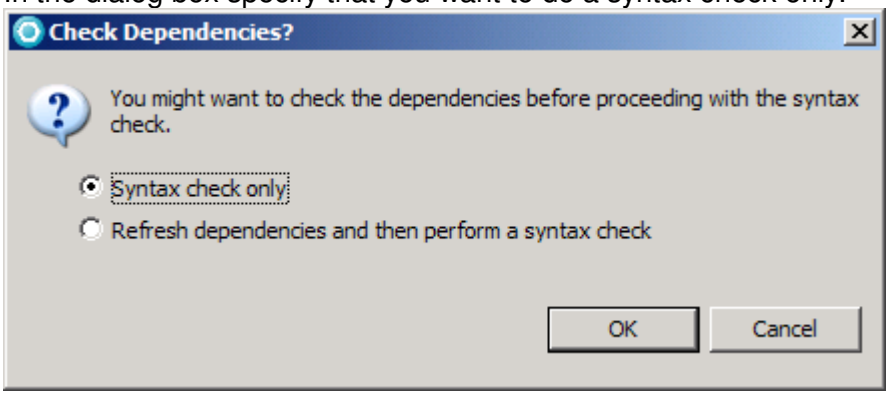
126. If you do not like to have the real time syntax check activated, you might change the settings for the Cobol Parser Settings in Window → Preferences → and in the navigation tree LPEX Editor | System z LPEX Editor | Cobol LPEX Editor



The next steps describe the explicit local syntax check.

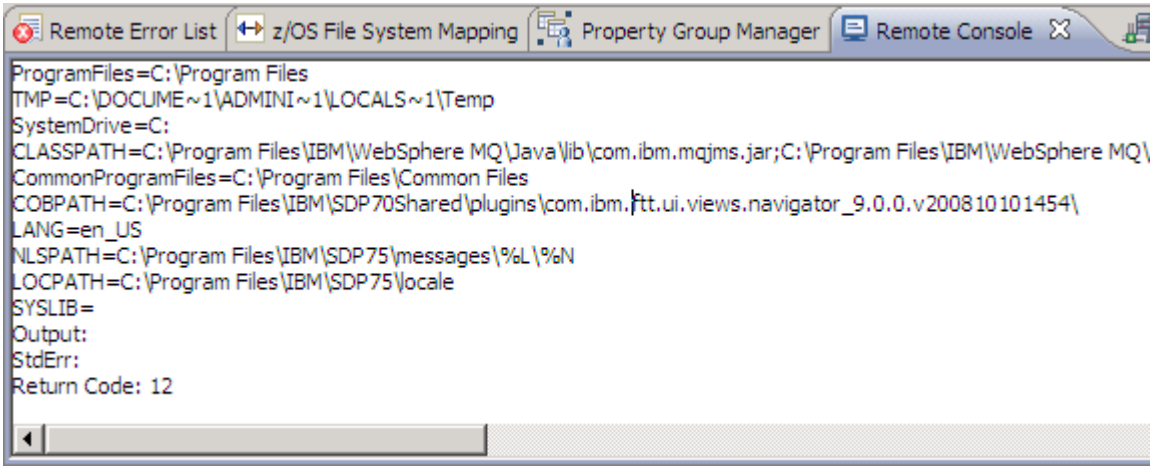
127. Right click anywhere within the code or right click on HELLOW.cbl in your z/OS Projects View and select Syntax Check → Local.

128. In the dialog box specify that you want to do a syntax check only.

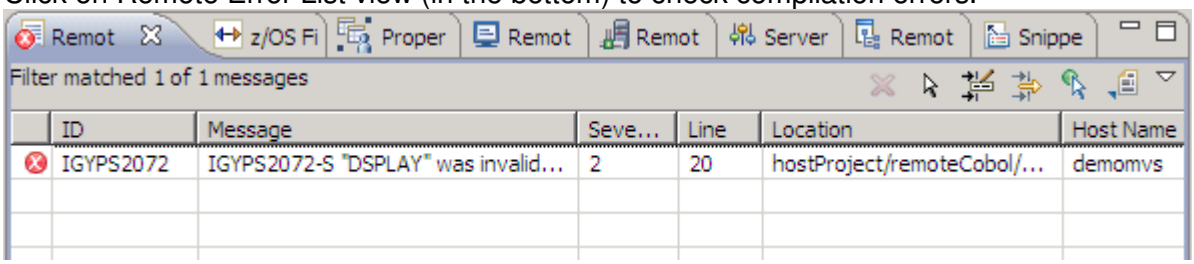


Note: For a local syntax check local resources will be used. This means you can save CPU by performing local tasks.

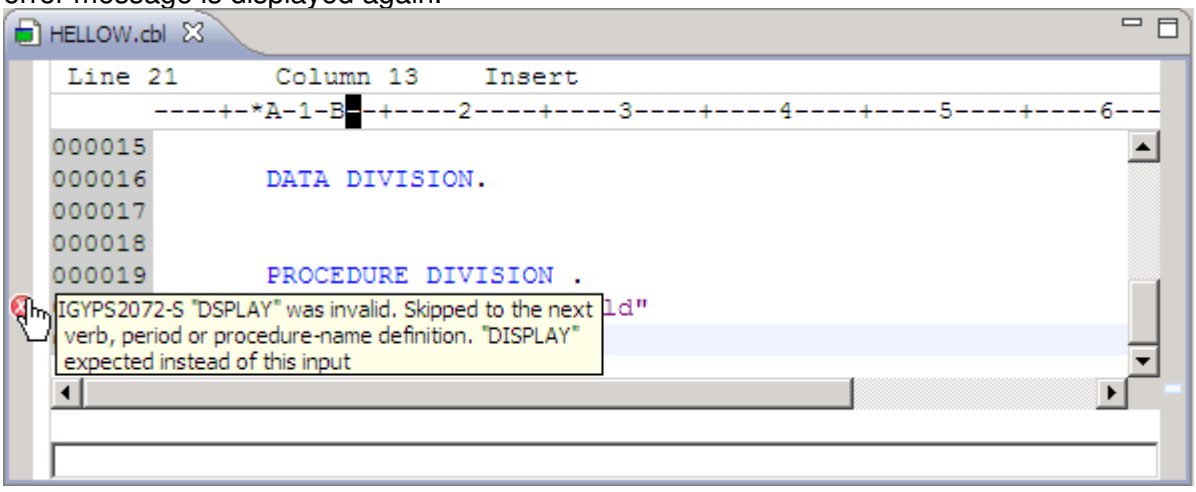
129. The TPF Toolkit Console will provide Feedback about the compile process. It should return a reason code of 12.



130. Click on Remote Error List view (in the bottom) to check compilation errors.

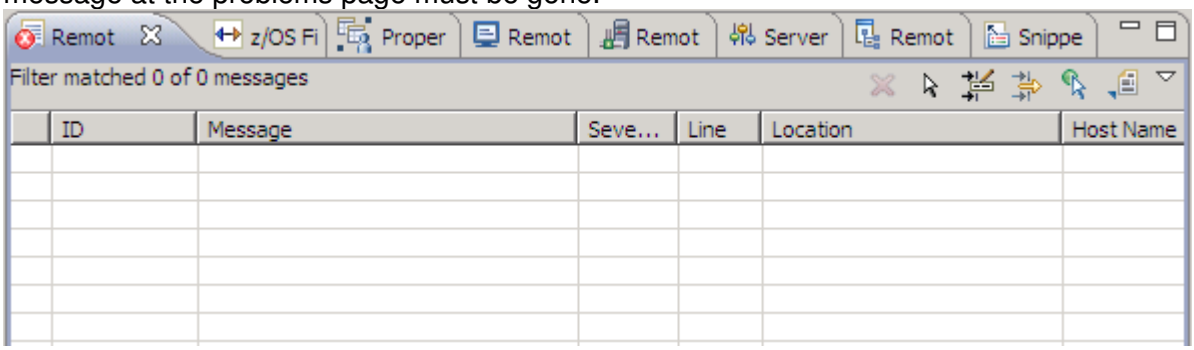


131. Double click on the error message. This should bring you to the editor. Note that a red mark was placed next to the error. If you use your mouse to hover over this mark the error message is displayed again.



132. Fix that returning the old version that had the correct DISPLAY statement..
Click CTRL + F4 or CRTL + w to close the editor..

133. Perform another Syntax Check to ensure that the compilation is now clean. The error message at the problems page must be gone.



134. Close all opened editors if still opened. (CTRL + F4)

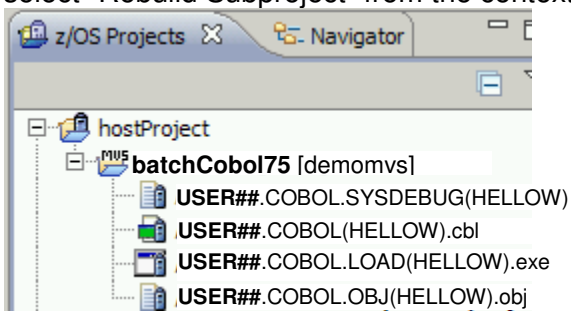
3.6 Compile/link/execute the remote COBOL program

In this part you will learn how to compile your programs, link them and execute them on z/OS.

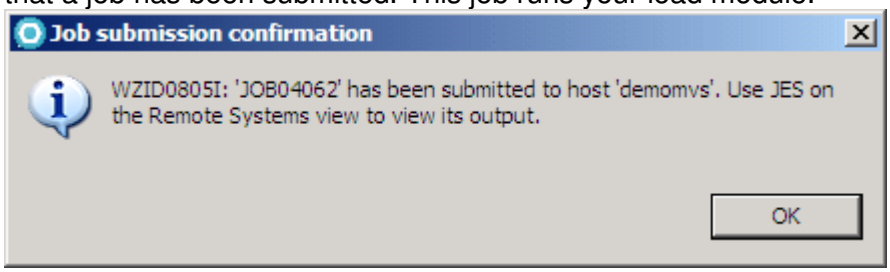
3.6.1 Using Project Build

The Project Build Feature will build all Members contained in your project automatically. At the moment, your whole Cobol dataset is contained in the project, so for this section we will remove it and add the Cobol member only.

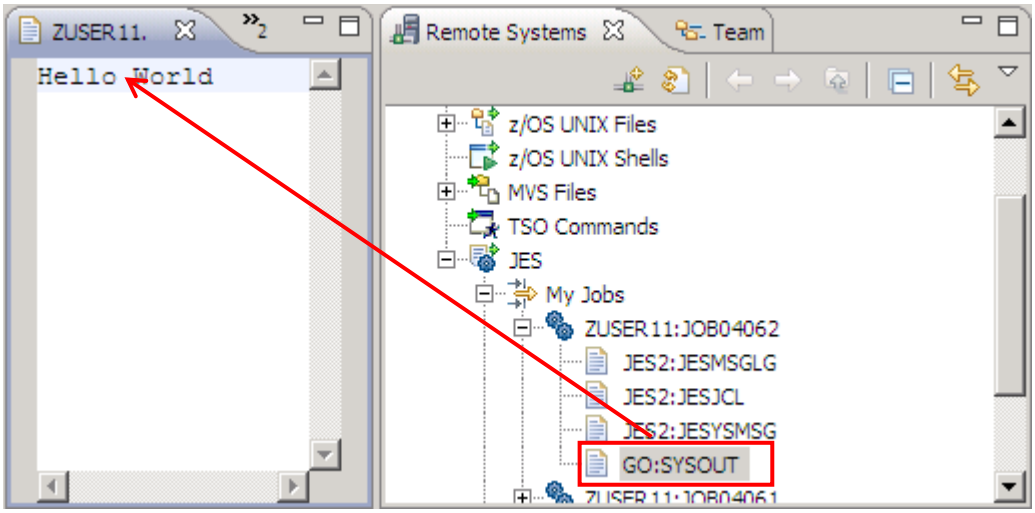
135. Go back to the "z/OS Projects" View, right click your remoteCOBOL subproject and select "Rebuild Subproject" from the context menu. The result will look similar to



136. To run your project, right click your load module USER##.COBOL.LOAD(HELLOW).exe and select "Run" application from the context menu. A popup window will inform you that a job has been submitted. This job runs your load module.



137. To review the output go to the "Remote Systems" View and expand JES and an appropriate filter (at least the "My Jobs" filter should display your results) to review your job output.



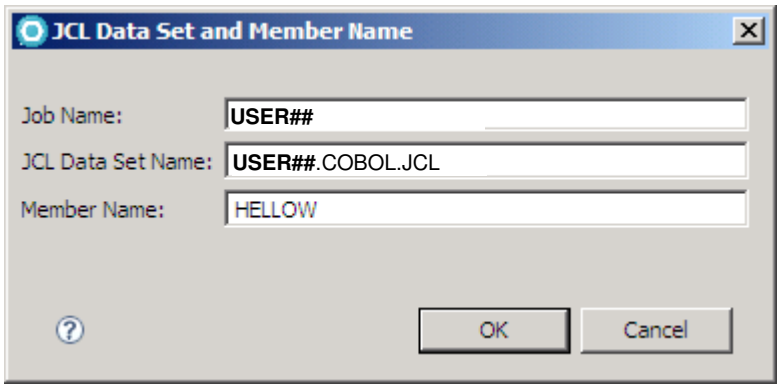
3.6.2 Using JCL Generation

You will use a feature called JCL generation, where the properties of your Remote z/OS project are used to generate JCL for Compile only, Compile and Link or Compile, Link and Go.
We are using this feature for demonstration purposes. In real world of course you will either use you company's provided compilation procedures or take advantage of a source code management system like SCLM or Endeavor.

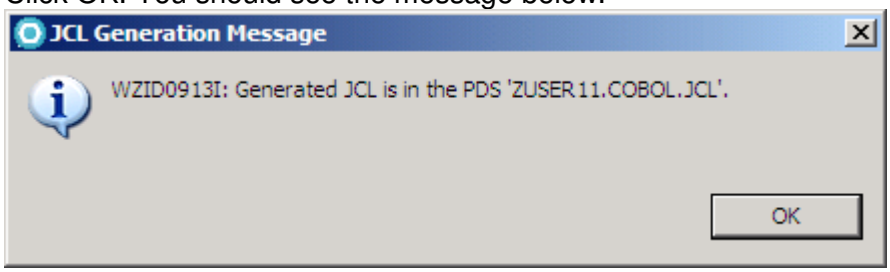
Now that you have a successful syntax check of your COBOL program, you can generate the JCL (Job Control Language) that will be used to create the executable on your z/OS system.

138. Using the z/OS projects View, Right-click on HELLOW.cbl and select Generate JCL → For Compile Link Go.

139. On the JCL Data Set and Member Name window, notice that the JCL Data Set Name is set to the value you specified for your project settings.

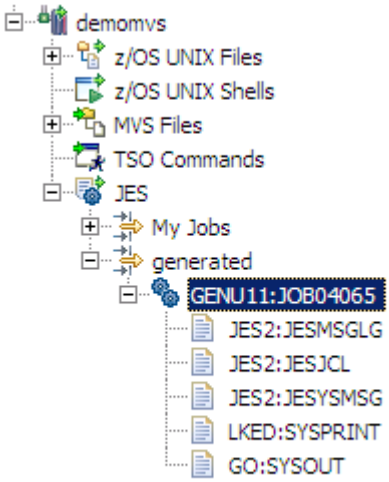


140. Click OK. You should see the message below.

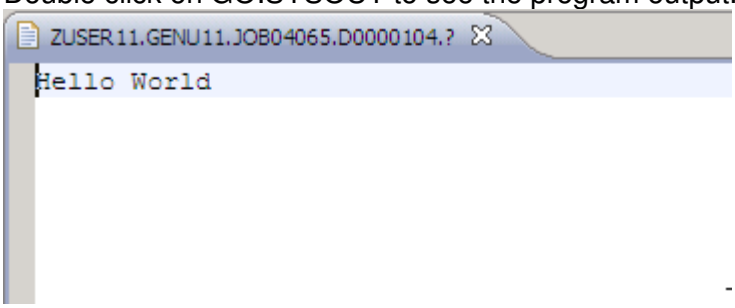


141. Go to your z/OS Projects view and you will see that HELLOW.jcl was generated.

142. Submit your JCL (Right click → Submit), go to your Remote Systems View and expand the Job under JES.



143. Double click on GO:SYSOUT to see the program output.



3.6.3 Using Menu Manager

You can use a feature called Menu Manager to parameterize JCL with variables from your workbench. Please refer to the separate Menu Manager Lab.

3.7 Debugging Cobol

We will now have a look at a little more complex program and execute it in debug mode. The member REGI0A.cbl has been copied to your Cobol dataset.
To execute in debug mode, we have 3 options:

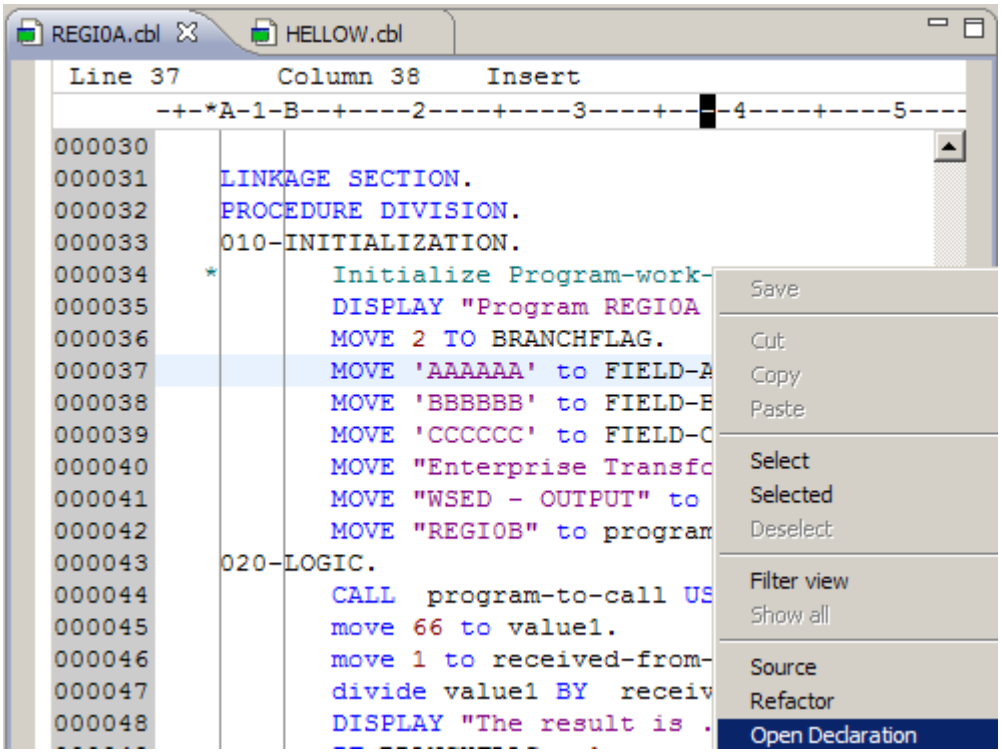
- Modified property group for the JCL generator
- Right click menu of a load module within a MVS project
- Exits

144. Either create a new MVS subproject or empty your existing project (careful, don't delete, select "remove from subproject" from the right click context menu) and add REGI0A.

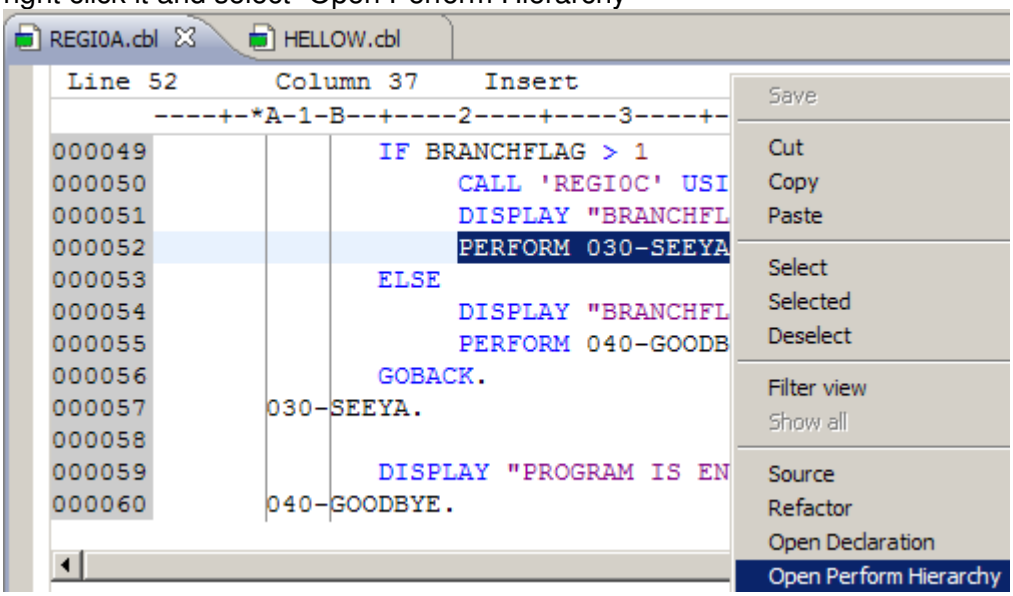
3.7.1 Some more editor goodies

As REGI0A is a little more complex we can explore some more editor functions.

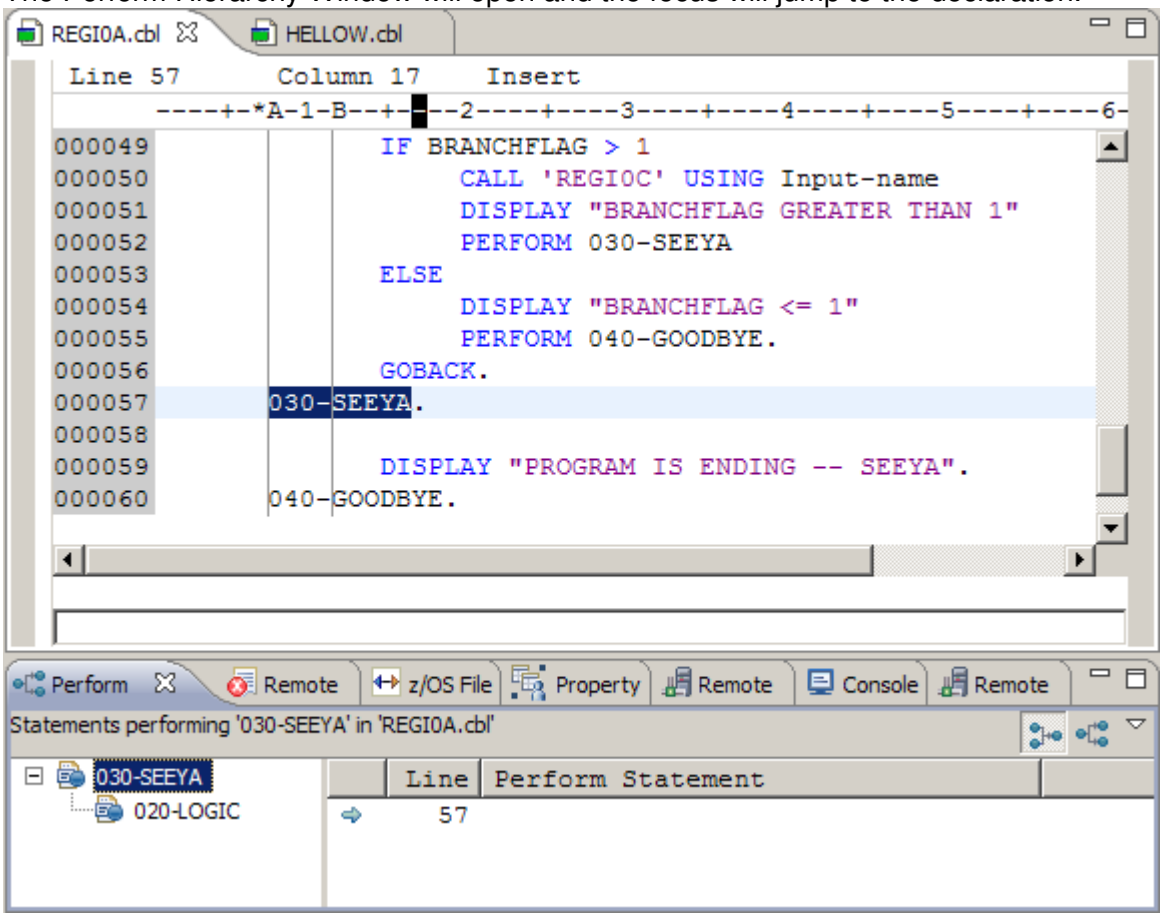
145. Right click on the variable "FIELD-A" and select "Open Declaration". The focus will jump to the declaration.



146. Mark the line saying
PERFORM 030-SEEYA
right click it and select “Open Perform Hierarchy”



147. The Perform Hierarchy Window will open and the focus will jump to the declaration.



148. Click on the  buttons to toggle between Performer and Performee Hierarchy

3.7.2 Modified property group for the JCL generator

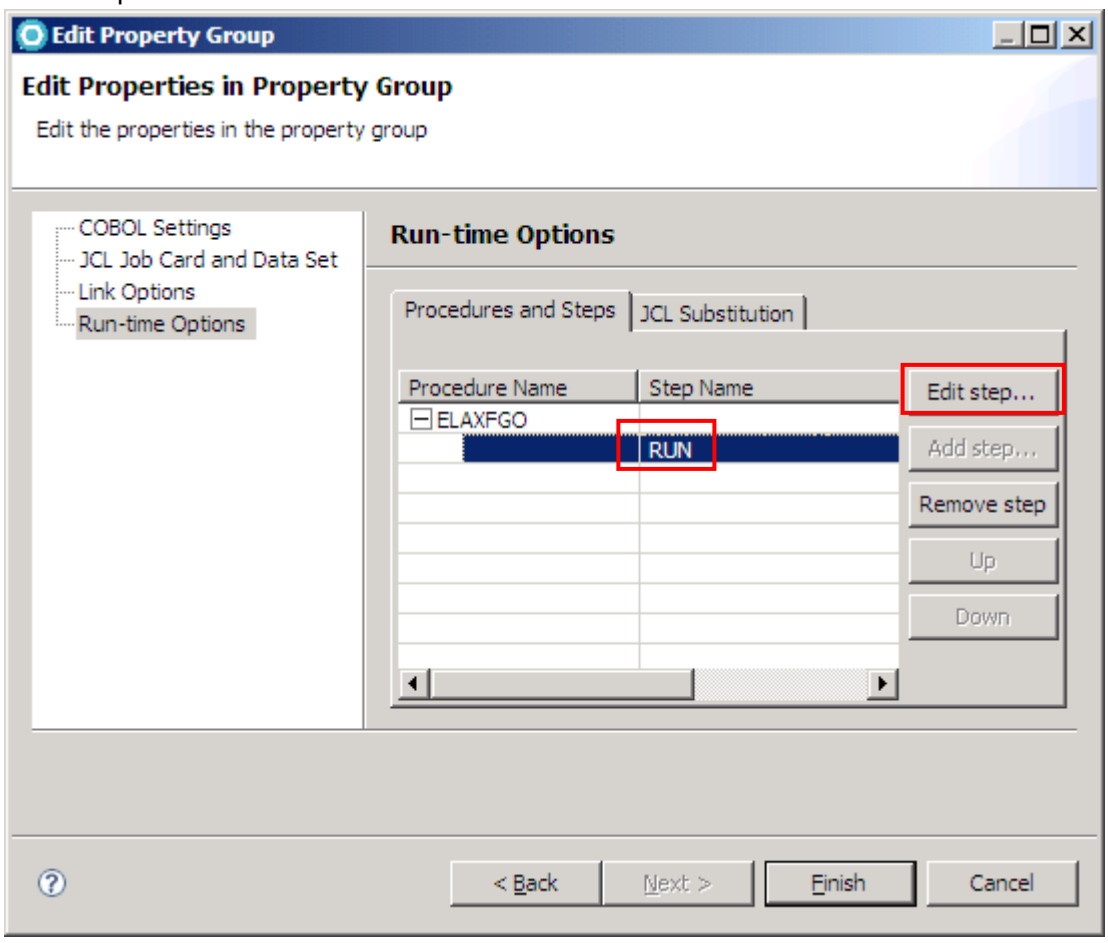
Your generated JCL can be modified to initialize debugging. Instead of modifying the parameters, we will create a second Property Group for Debugging.

149. Navigate to the Property Group Manager and right-click on the existing Property Group batchCobol75 and select Copy…

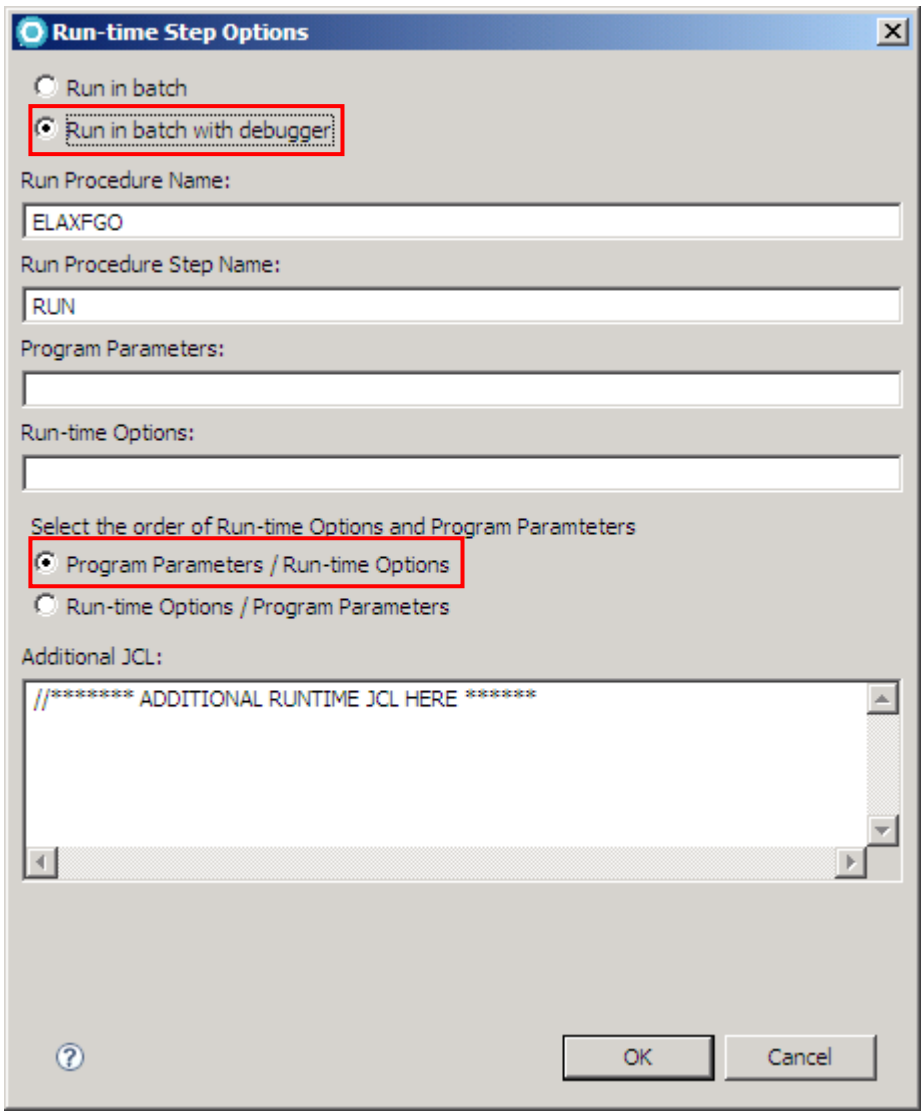
150. Now right-click the new Property Group "Copy of batchCobol" and open the edit panels. Before proceeding rename the Property Group to a more meaningful name like batchCobol75 Debugging. Hit Next.

All Properties are now identical to the template. We now only modify the parameters for debugging.

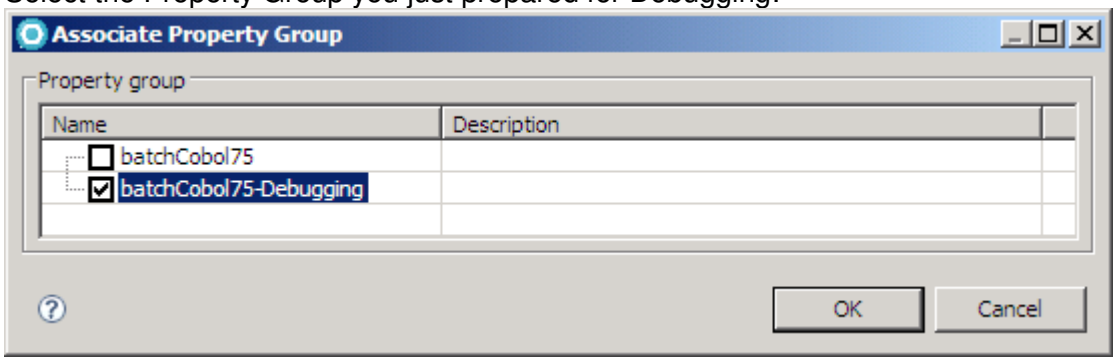
151. On the Run-time Options panel, expand the + next to ELAXFGO, select RUN and click "Edit Step".



152. On the Step Options Panel, select “Run in batch with debugger” and make sure that “Program Parameters / Run-time Options” is selected as order. Click OK. And then Finish to save changes.

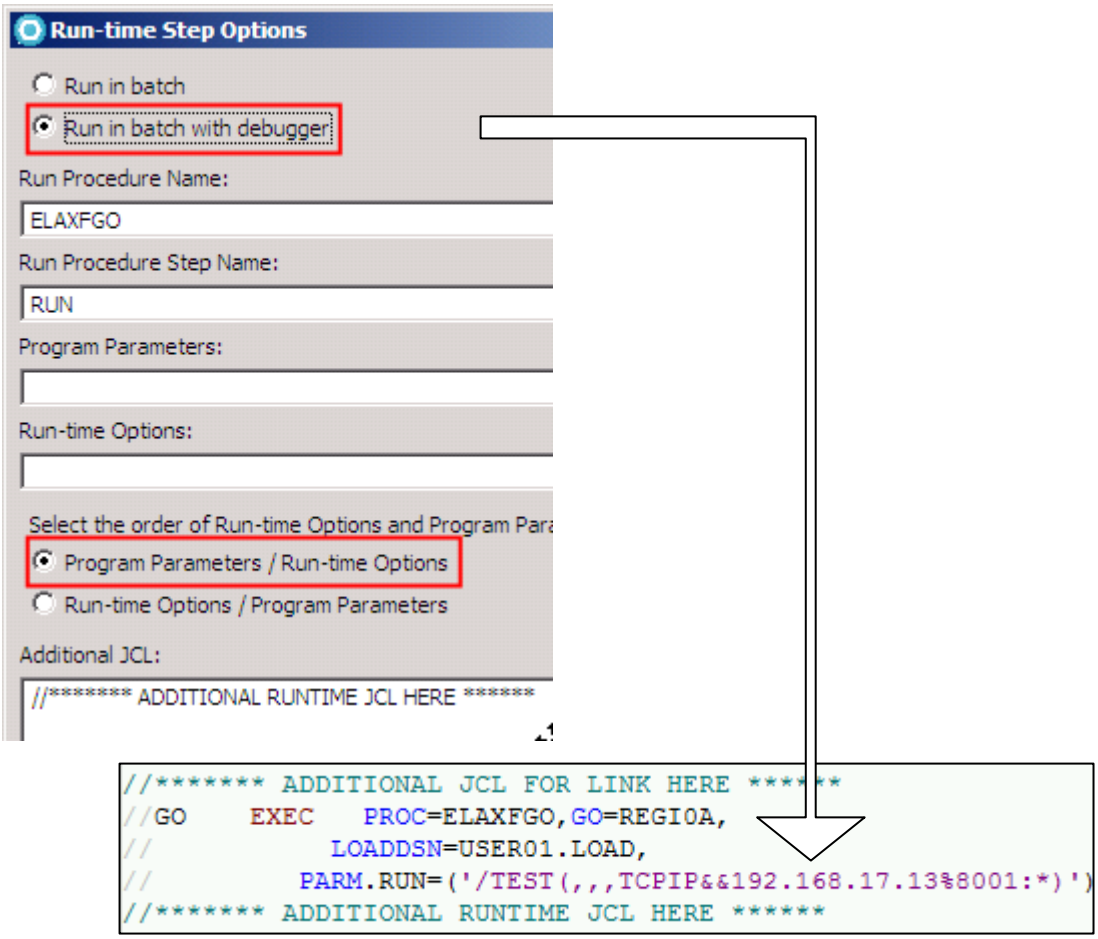


153. Now do a right click on REGI0A.cbl and select Property Group → Associate Property. Select the Property Group you just prepared for Debugging:



154. To generate the JCL right click REGI0A.cbl and select Generate JCL → For Compile, Link and Go.

155. Open the generated JCL and scroll to the end of the JCL to look at the runtime options. RDz inserted a TEST runtime parameter specifying your IP address. This was done because you asked for remote debugging in the Run-time Step Options.



156. Submit the Job. After a moment, you are asked, if you want to switch to the Debug Perspective

You can now either cancel debugging to try the other options or you can directly jump to chapter “3.7 Debugging Cobol” on page 64 and try the other options later.

3.7.3 Right Click Menu

157. Initiate a project build on the project that you added REGI0A to. Your load member should be available now (USER##.COBOL.LOAD(REGI0A))

158. From your project, right click the load member and select “Debug Application” from the context menu.

You can now either cancel debugging to try the other options or you can directly jump to the debugging instructions and try the other options later.

3.7.4 User Exits

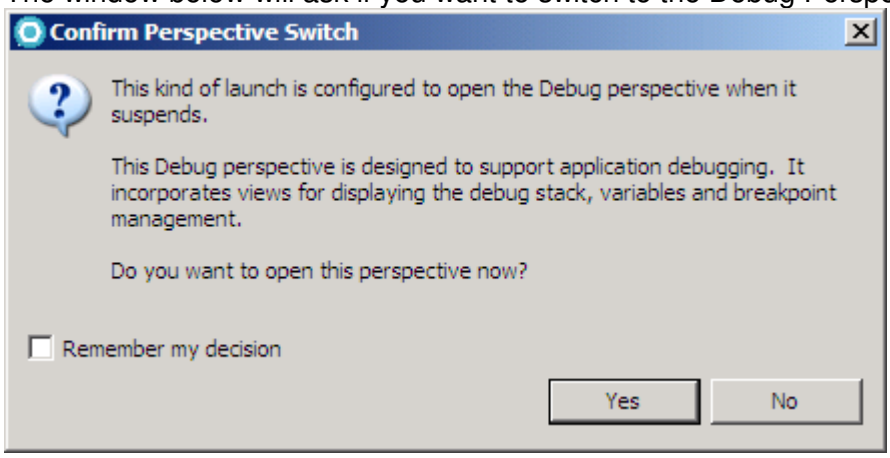
This feature is currently not available.

3.7.5 Using Menu Manager

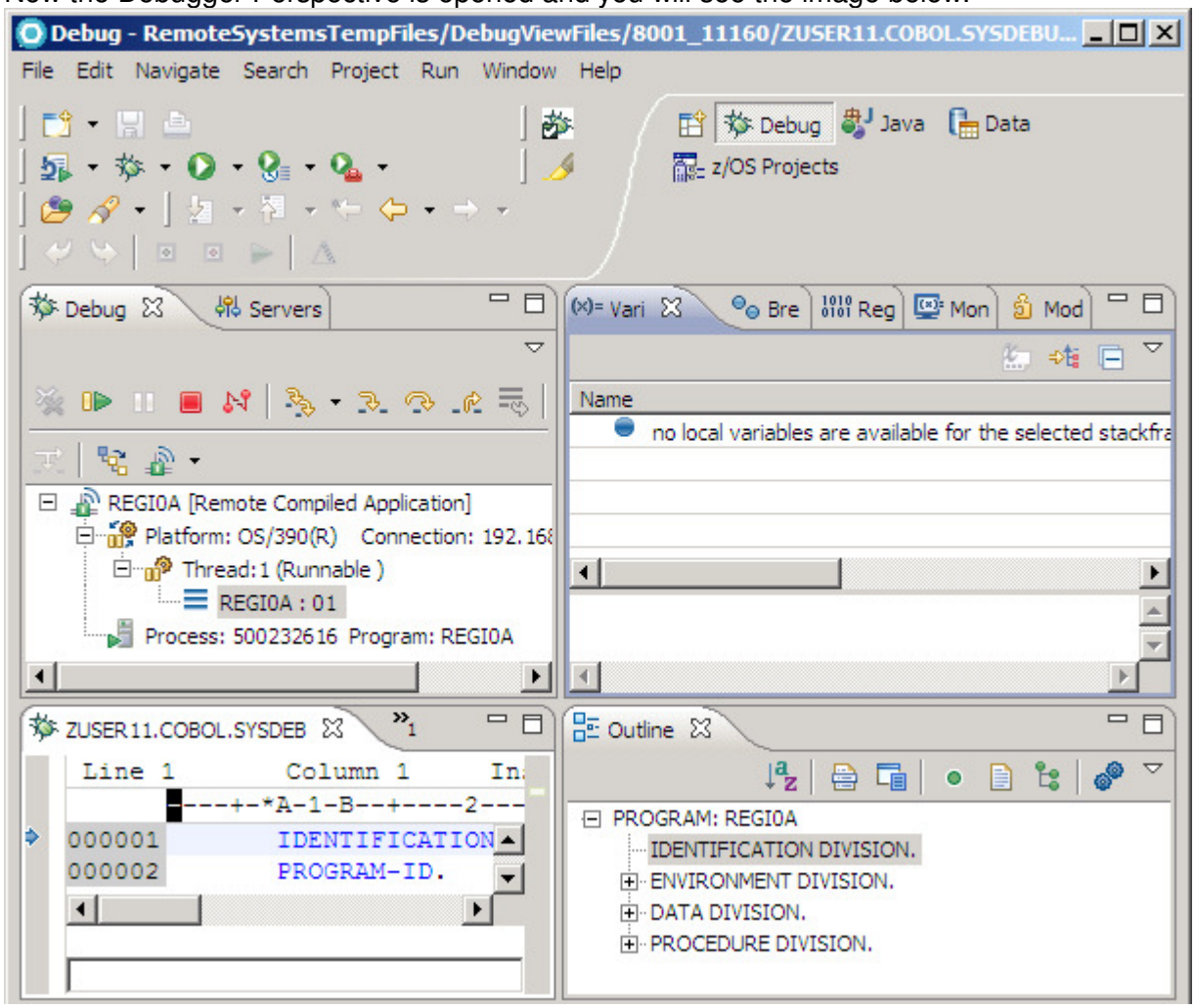
You can use a feature called Menu Manager to parameterize JCL with variables from your workbench. Please refer to the separate Menu Manager Lab.

3.7.6 Debug

159. The window below will ask if you want to switch to the Debug Perspective. Click Yes.

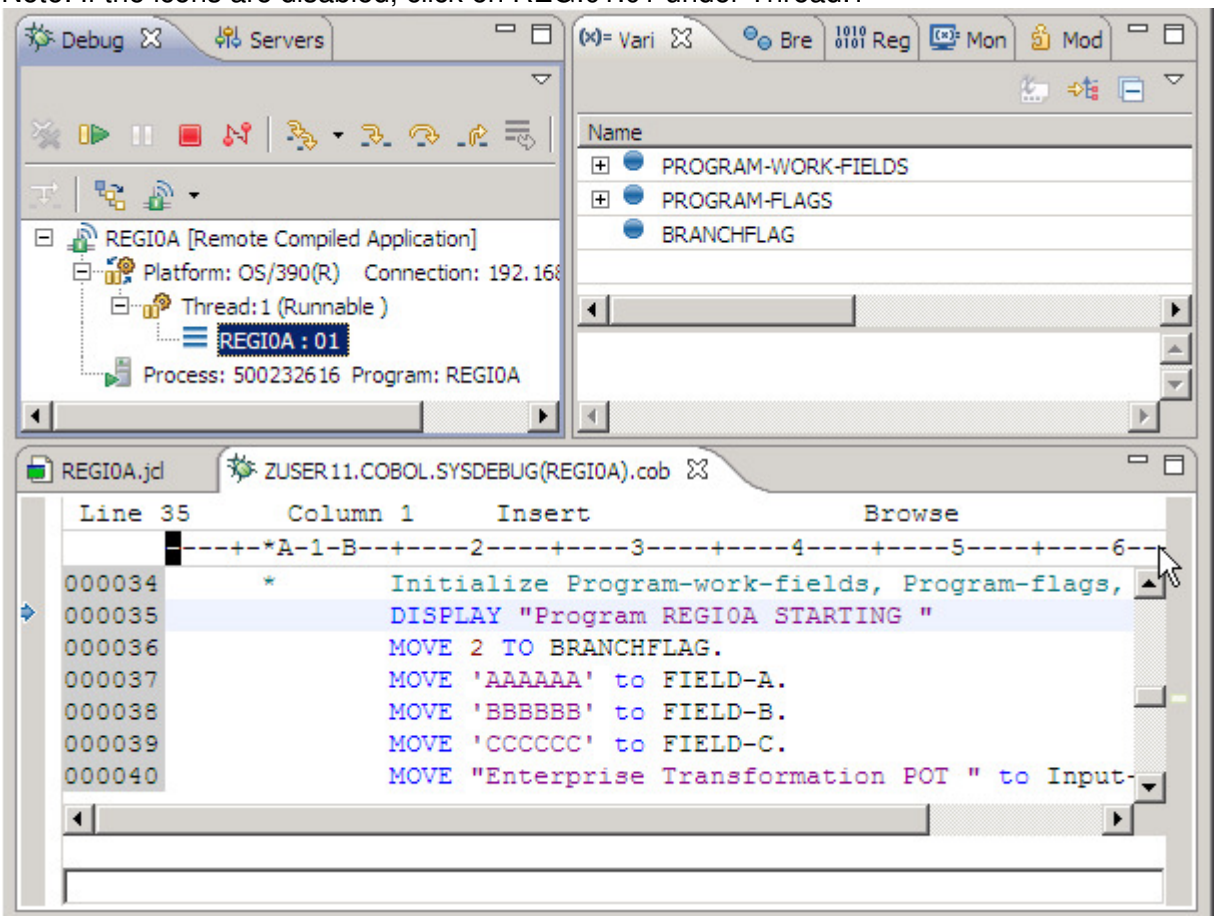


160. Now the Debugger Perspective is opened and you will see the image below.

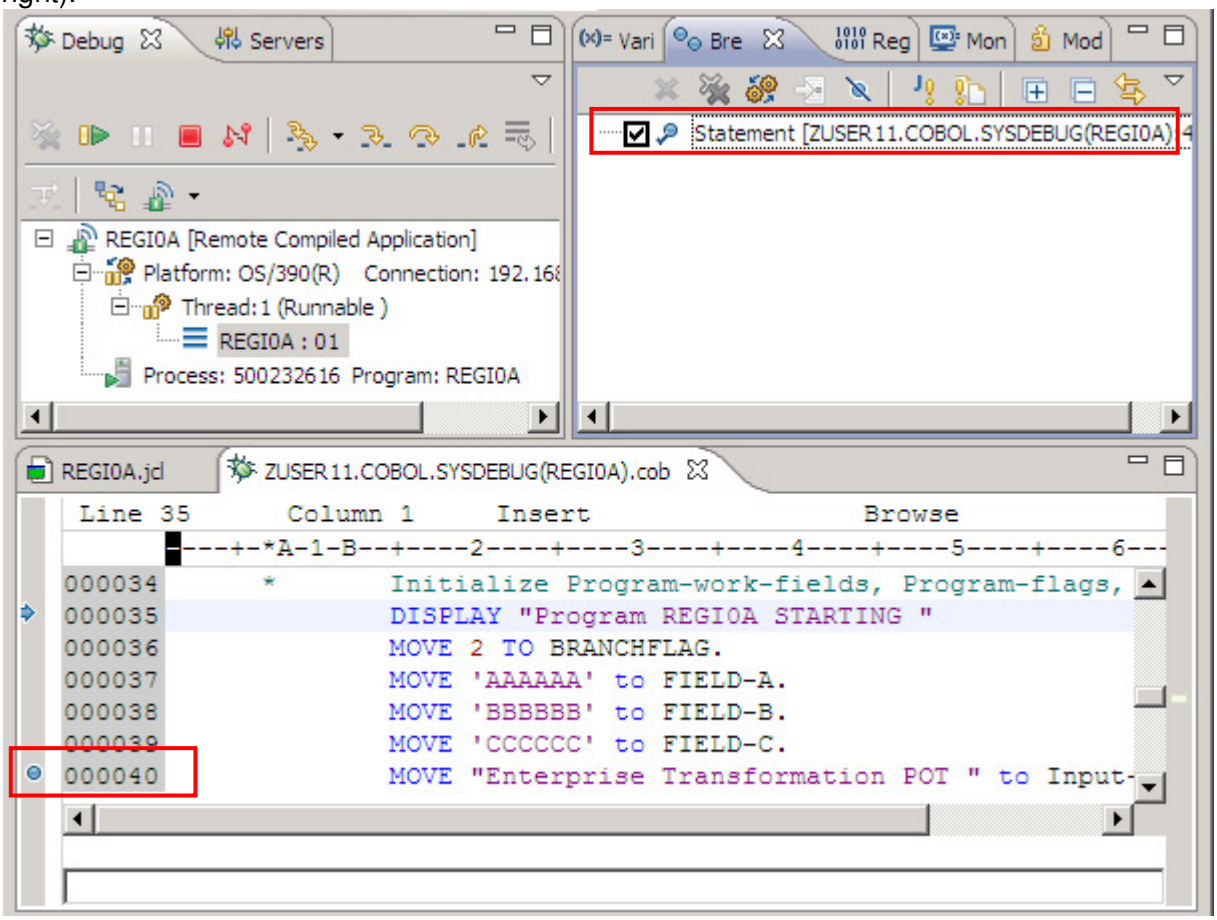


161. Close the outline view that is not be useful in this debugger.

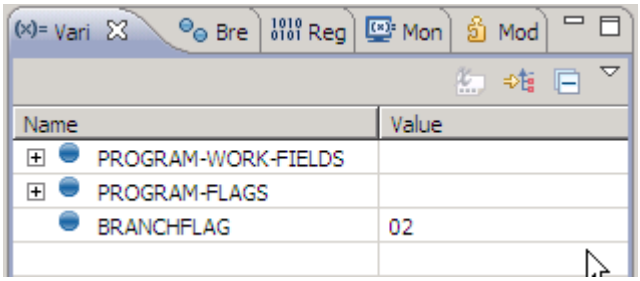
162. Click twice on the icon Step Over (or F6) located in the Debug view until you stop in the DISPLAY command:
Note: If the icons are disabled, click on REGI01:01 under Thread:1



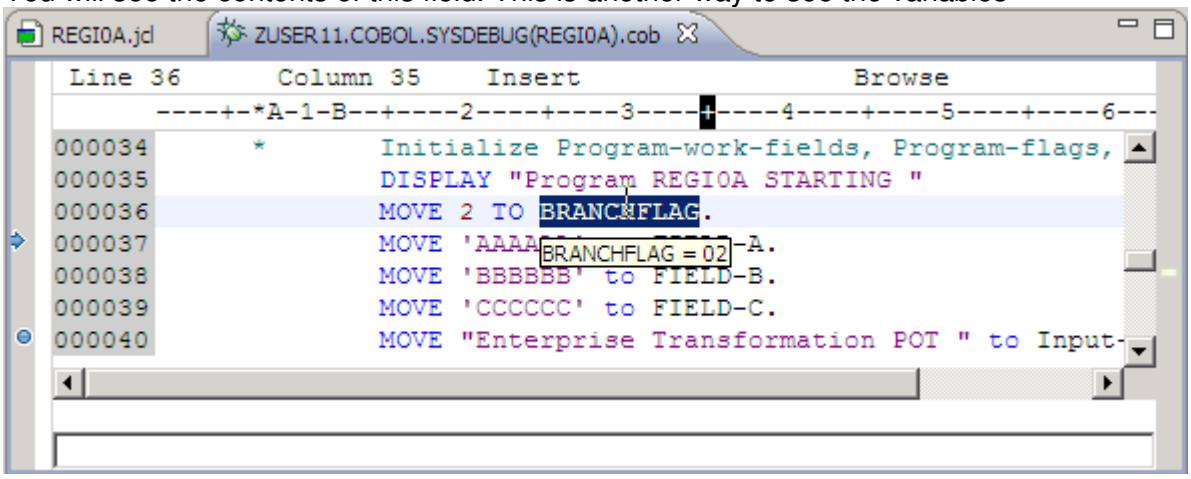
163. Double click to the left of line number 000040 to add a breakpoint. The symbol ( ) appears. Note that the breakpoint will also be added to the Breakpoints View (upper right).



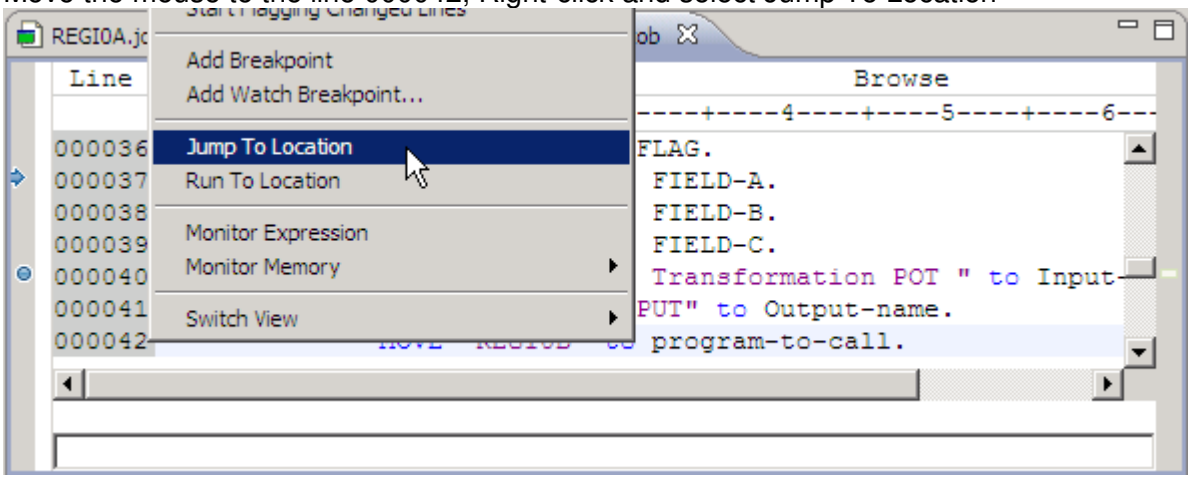
164. Click the Step Over button (or F6) twice.
Click on Variables view (Upper right) and you will then see the COBOL working-storage data items.



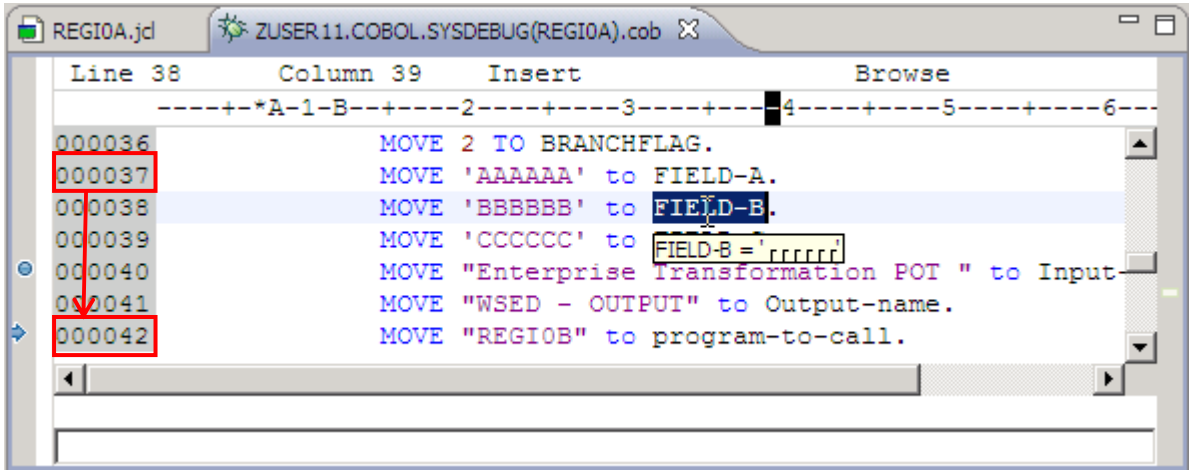
165. Move the mouse to the field BRANCHFLAG and wait 3 seconds or doubleclick on it. You will see the contents of this field. This is another way to see the variables



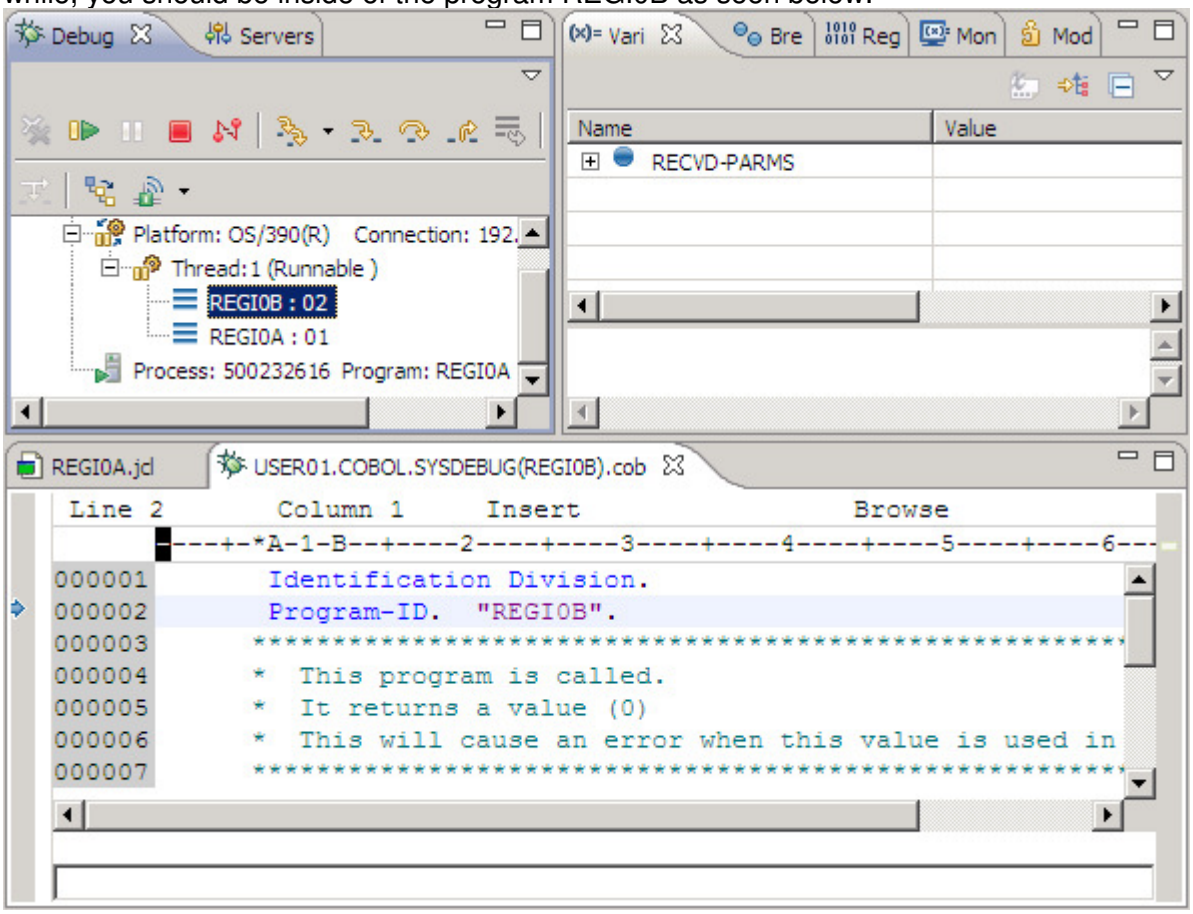
166. Move the mouse to the line 000042, Right-click and select Jump To Location



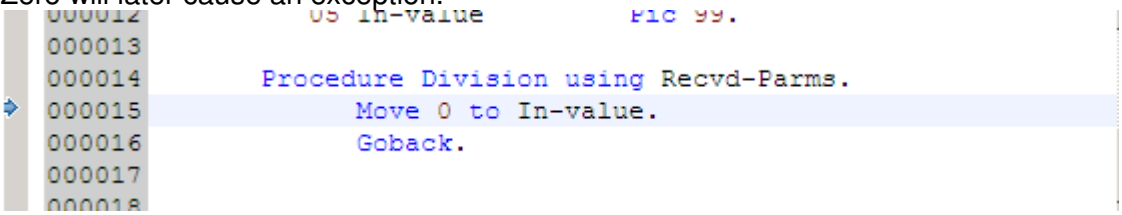
167. The program jumps from line 37 to 42 but did NOT execute the statements that were jumped. This can be verified just moving the mouse to one of the fields like FIELD-B. Also note that the breakpoint that you added did not work, since it was jumped.. See below.



168. Click STEP INTO (or F5) twice since we want to debug the Called program. After a while, you should be inside of the program REGI0B as seen below:



169. Click STEP INTO (or F5) again. As you see the program REGI0B just moves 0 to In-value and returns. This value will be used in a division. And guess what? Division by Zero will later cause an exception.

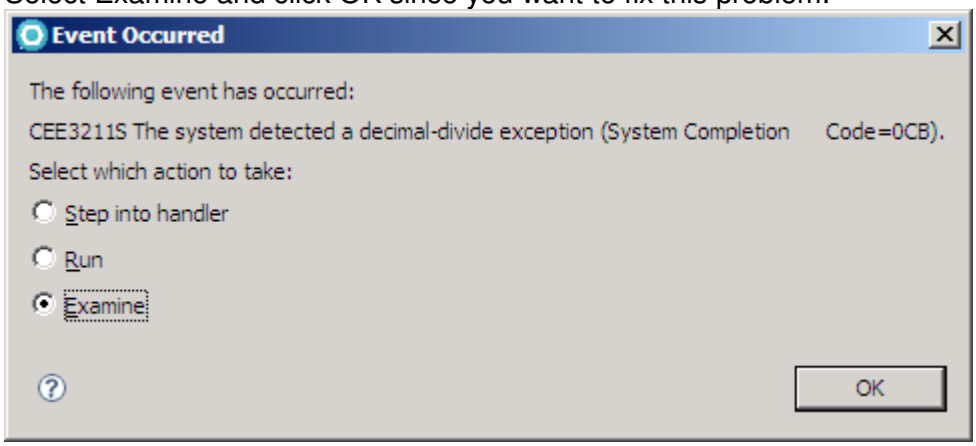


170. Now step over several time times until the cursor is positioned in line 46. We now want to produce a division by zero and jump over line 46 without executing line 46, which sets the RECEIVED-FROM-CALLED field again to 1. This is done only for the correctness of this program for our other labs.

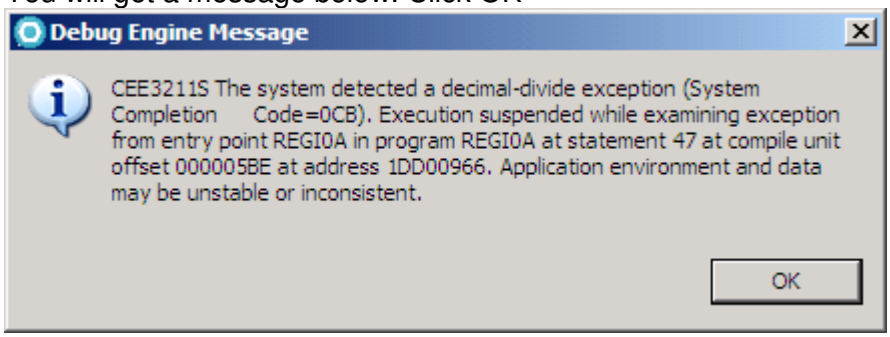
So position the curser to line 47 and hit Jump to Location from the context menu.

171. Just use resume (F8) until you get the exception due the division by zero (0CB)

172. Select Examine and click OK since you want to fix this problem:



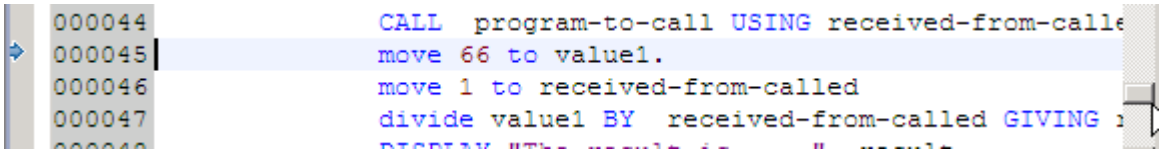
173. You will get a message below. Click OK



174. Move the mouse to the line 45 (will need to scroll back) and press the left mouse button to position the cursor there.

175. Now Right-click on line 45 and select Jump to Location, since we want to execute the divide again. Depending on the version that you are using you might need to do that twice.

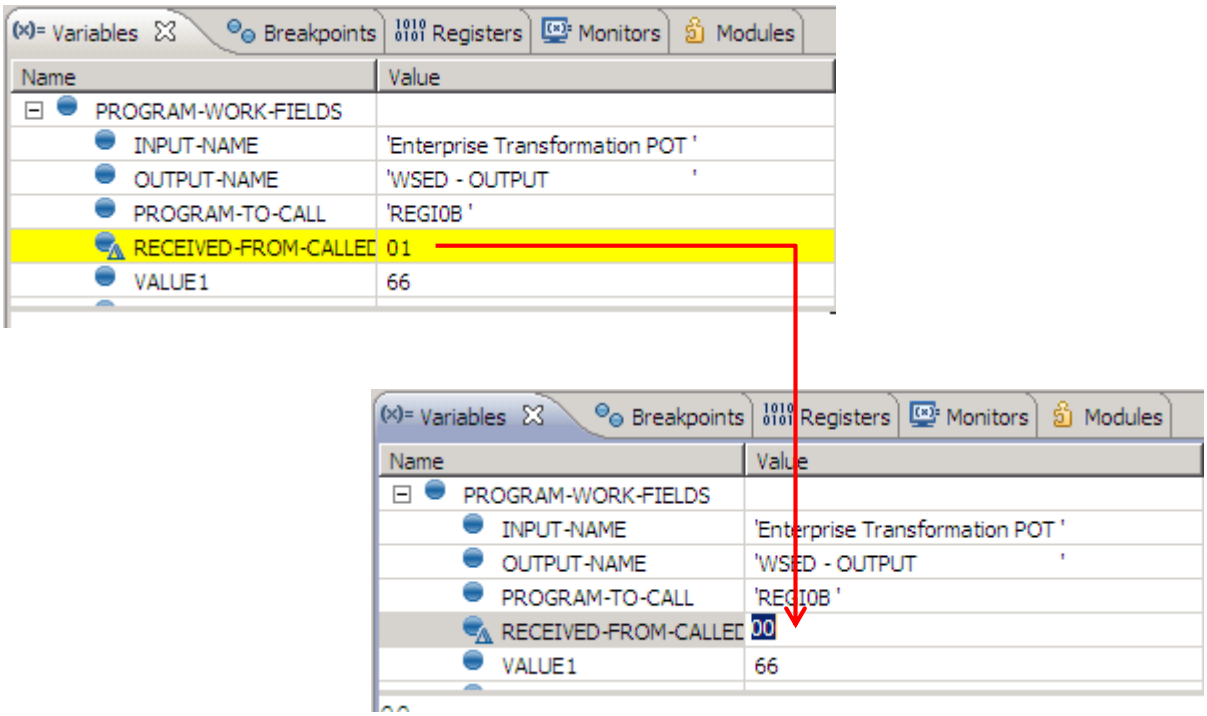
176. You will be now on the line 45 as shown below
:



177. Now step over twice (F6), to get into line 47.

178. Using the Variables view, expand PROGRAM-WORK-FIELDS, navigate to RECEIVED-FROM-CALLED field and click on its value 01, since we executed the Move in line 46.

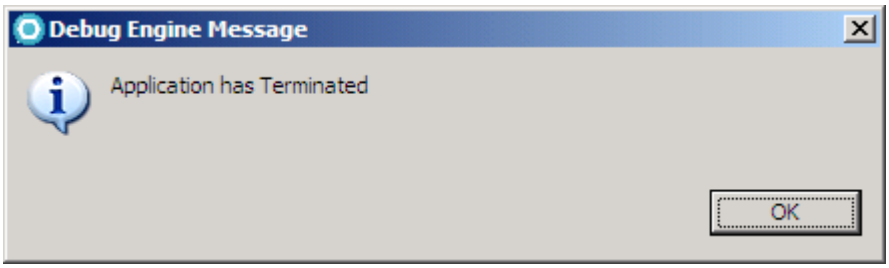
     The value will become editable. You can change it to a desirable number; you also can produce a division by zero.



179. Depending of your modification, you might now correct it a second time, like described in the last steps. If you have no division by zero, proceed with the next step.

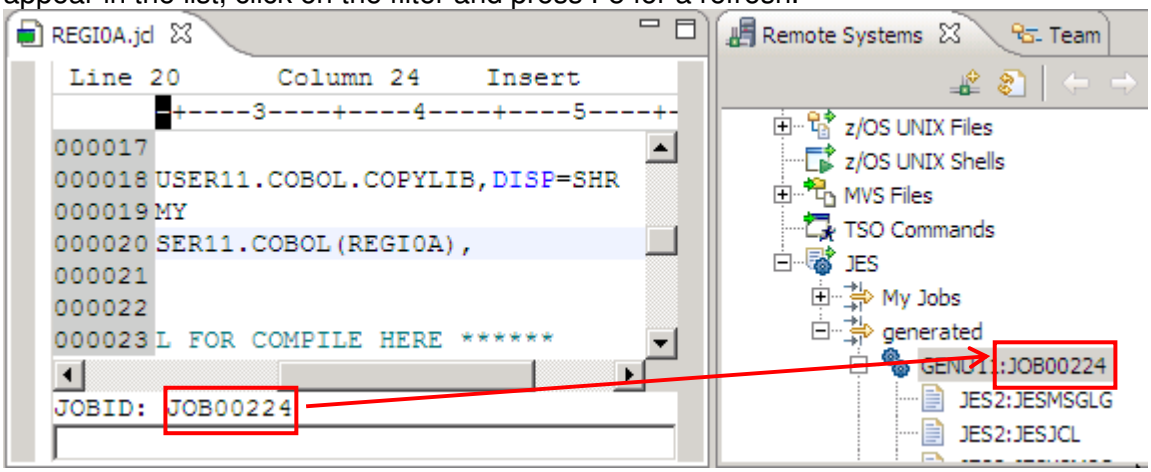
180. Click STEP INTO (or F5) until you see the next called program REGI0C being called

181. Click on Step Over (F6) and (F8) when you are satisfied so the program will execute until the end and click OK.

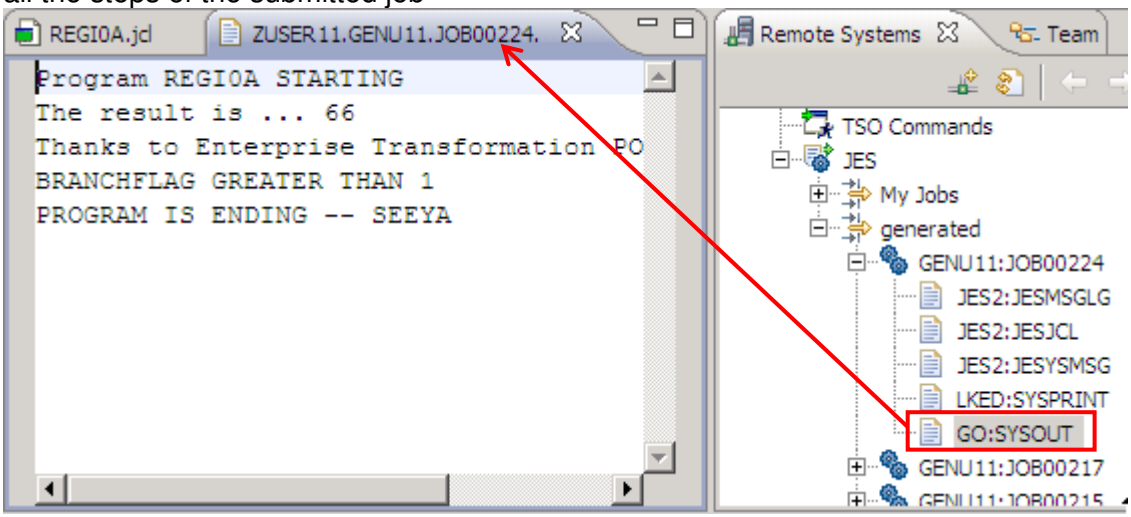


182. Using the Remote Systems view locate the node demomvs and expand the JES node and either the general filter “My Jobs” or the filter “generated” which has been created in a previous step to specifically display your GEN* jobs.

183. You will see the job that you had submitted as the example below. If your job does not appear in the list, click on the filter and press F5 for a refresh.



184. Expand the job clicking on the + sign and double click in the step GO, you can also see all the steps of the submitted job



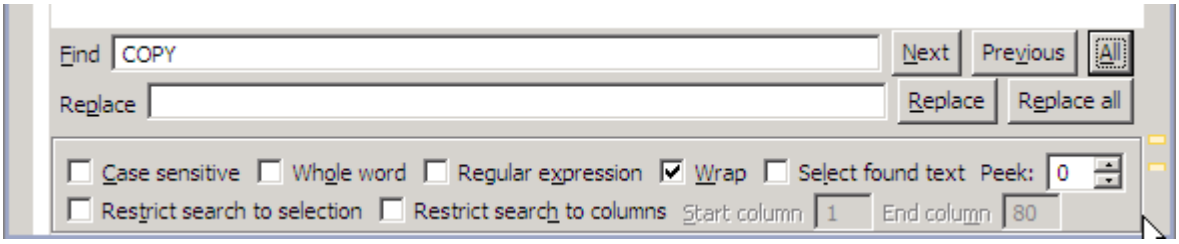
185. Close the editor (CTRL +F4)

186. Select the all your jobs (use CTRL key), right-click and select Purge. All job listing will be purged if you have authorization for.

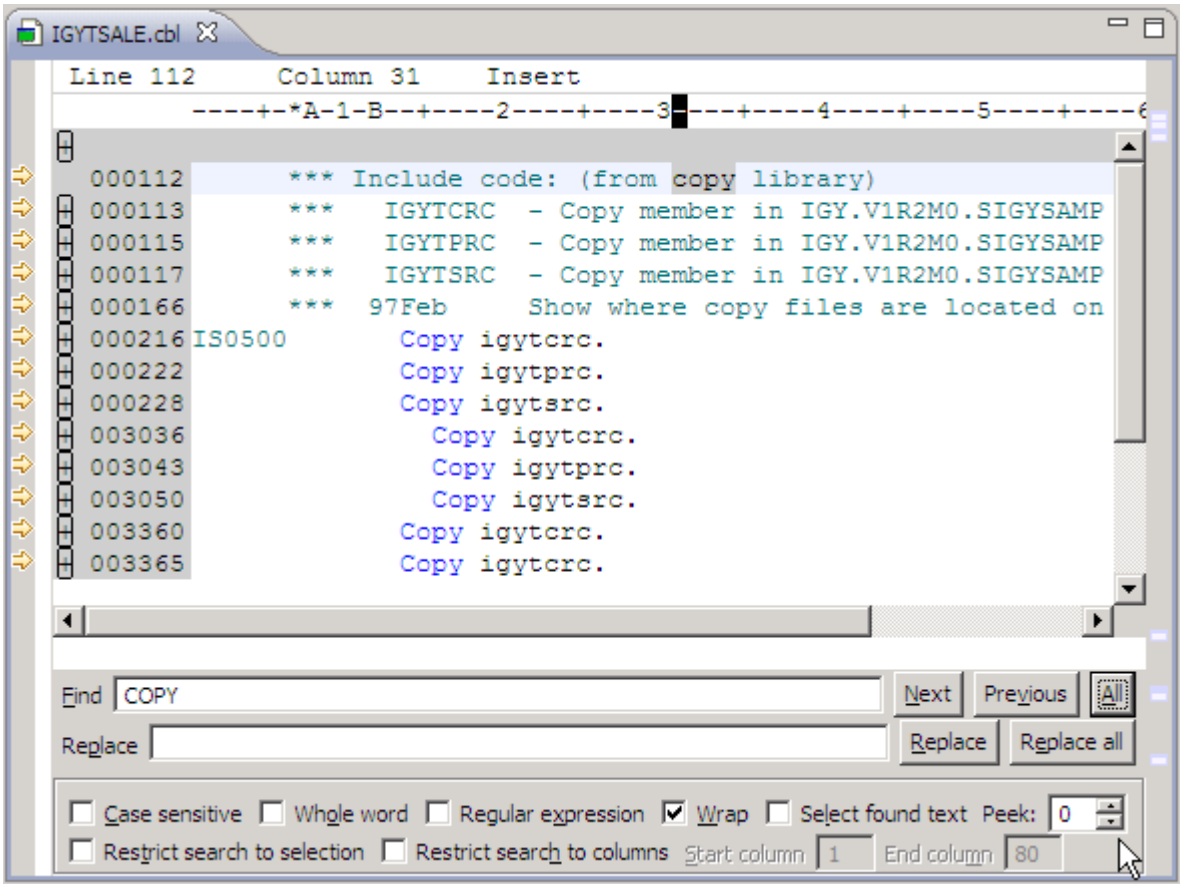
3.8 Copybook expansion and dependency check

187. Add IGYTSALE.cbl to your batch Cobol project (this has been copied to your Cobol dataset) and open it from there.

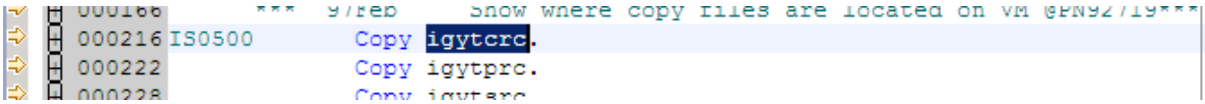
188. With the editing window of IGYTSALE.cbl active, press Ctrl +F to bring up the Find/Replace dialog (shown below). Enter copy in the Find field and click the All button.



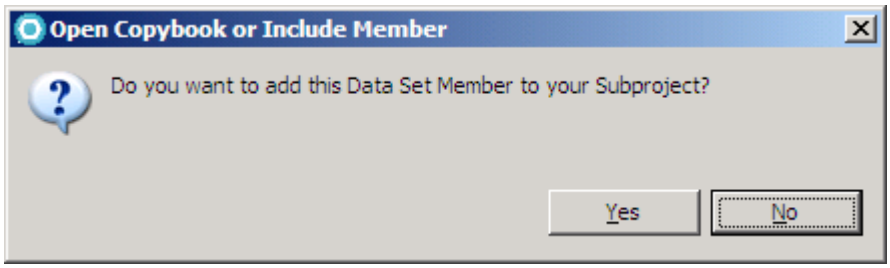
189. You should see something similar to the following. All lines with copy will be shown.



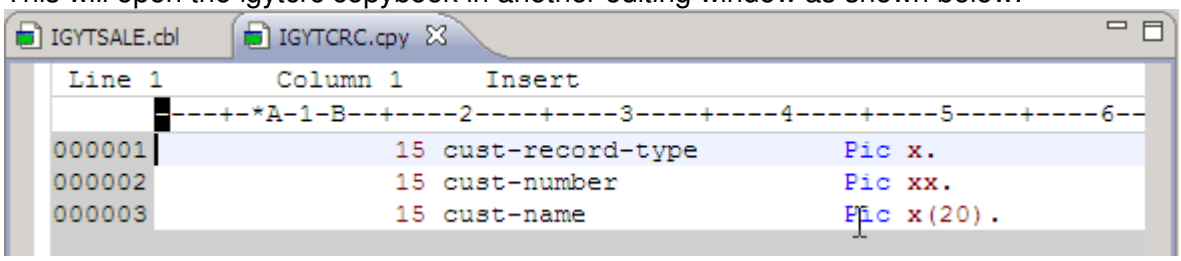
190. Highlight the copybook name igytcrc (line 216), press mouse button 2, and select the Open Copy Member action.



191. A window will open and ask if you want to add this Data Set Member to your Project. Click No

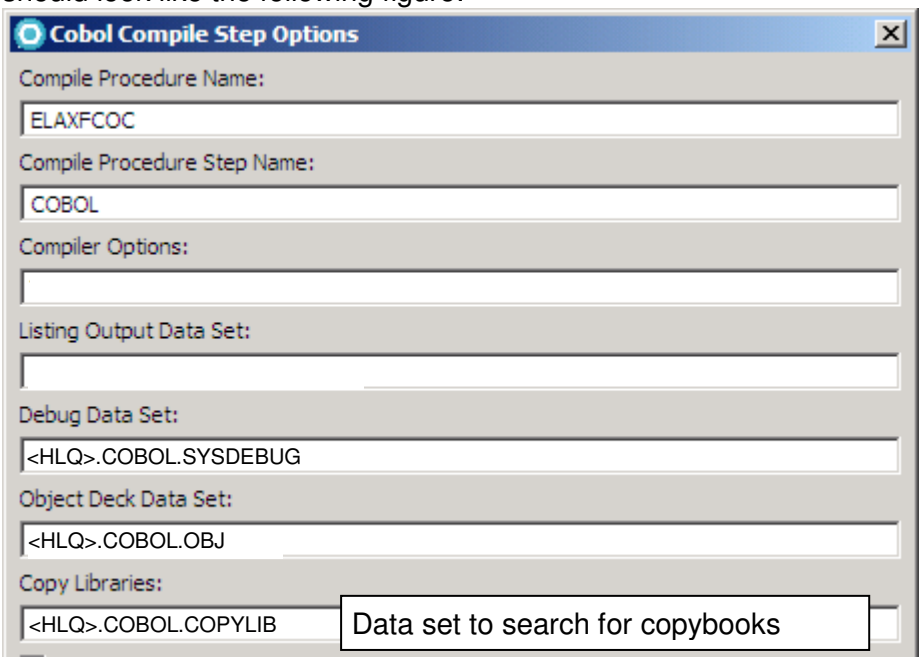


192. This will open the igytcrc copybook in another editing window as shown below:



Note that the copybook is shown even that is NOT part our MVS Project. Usually there is no sense to add copybook libraries to MVS projects, since they are used for many projects, unless they also need updates.

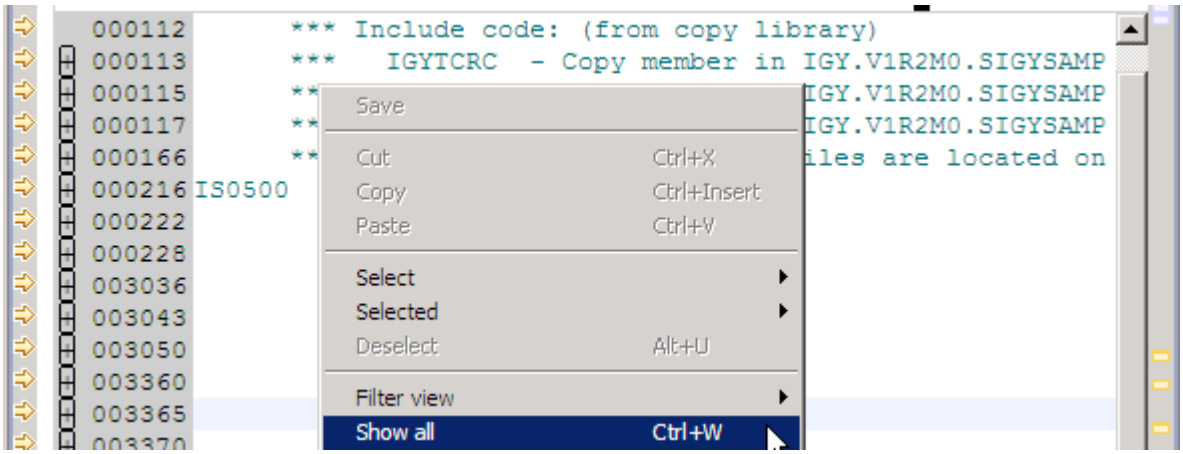
Copybooks are resolved based on the value of Copy Libraries in the COBOL Settings for the MVS project. The COBOL Settings of the property group of your remoteCOBOL MVS project should look like the following figure:



In this example, the copybook will be resolved (expanded) from USER##.COBOL.COPYLIB. You can specify multiple copy libraries, each separated by a space. This is equivalent to what on MVS is termed concatenation, in effect specifying the search order for copybook resolution/expansion.

193. Close the IGYTCRC.cpy editor.

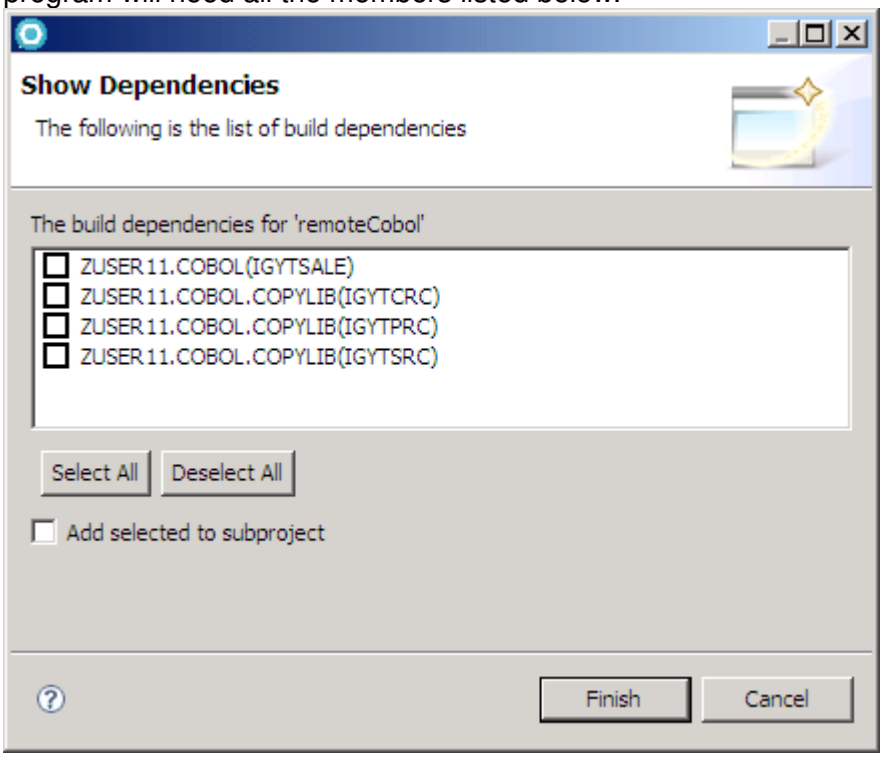
194. Click anywhere on the IGYTSALE.cbl editor area to make it the active editor. Right click and select the Show all action.



This action removes the filtered view (currently showing all statements with the word copy) and displays all statements.
RDz has a feature named Show dependencies that is useful when you need to identify which components are required to compile the COBOL program. For example when working off-line you will need to identify all dependencies that are necessary for a clean compilation.

195. Right-click on IGYTSALE.cbl in the z/OS Projects view and select Show Dependencies.

196. The COBOL compiler will be used to find the dependencies. You will see that this program will need all the members listed below.



197. Click on Finish to close this dialog. (Do not select any member)