

LAB – COBOL Application Development in z/OS using RDz (90 - 120 minutes)

This lab will take you through the steps of using the z/OS Application Development component of Rational Developer for System z to work with remote systems. It will familiarize you with the z/OS Application Development environment. If the connection to the mainframe is available, you will define a remote z/OS system, set up a MVS project, edit and compile a COBOL application. The process would be the same for a PL/1 program.

Each time you see this symbol  it means that you have to “do” something on your computer – not merely read the document.

Tips!



1) If you want to have the lab in HTML format displayed in your browser, you can find it at the location: **C:\RDZ_POT_V7\HTML**

Then you can open the html files using a internet browser.

2) Most of the labs will be performed under VMware. So we will run 2 'Windows' on the same machine. This will cause some overhead and sometimes the performance will not be as good as if the program would be running in the native windows... So, please be patient

3) If you lost the “VMware full screen” either type **Ctrl + Alt + Enter** or use the VMware icon  (on top).

Overview of development tasks

To complete this tutorial you will perform the following tasks:

1. **Connect to a z/OS System:**

→ Prepare your Workspace to connect to the zOS system, defining a Remote System and connecting to it

2. **Allocate z/OS Data sets:**

→ Allocate and load assets required for this lab.

3. **Associate z/OS resources to properties**

→ Configure the system data sets names, Job names to be generated, etc...

4. **Send the COBOL program to the z/OS**

→ You will copy a COBOL program from your workstation to the z/OS

5. **Create a z/OS Project**

→ Specify which data sets you will use in this tutorial, specify properties, etc..

6. **Work with z/OS remote assets** – edit, syntax check, submit, execute and see the output.

7. **(Optional) Working offline using z/OS Projects**

Section 1 – Connect to a z/OS System

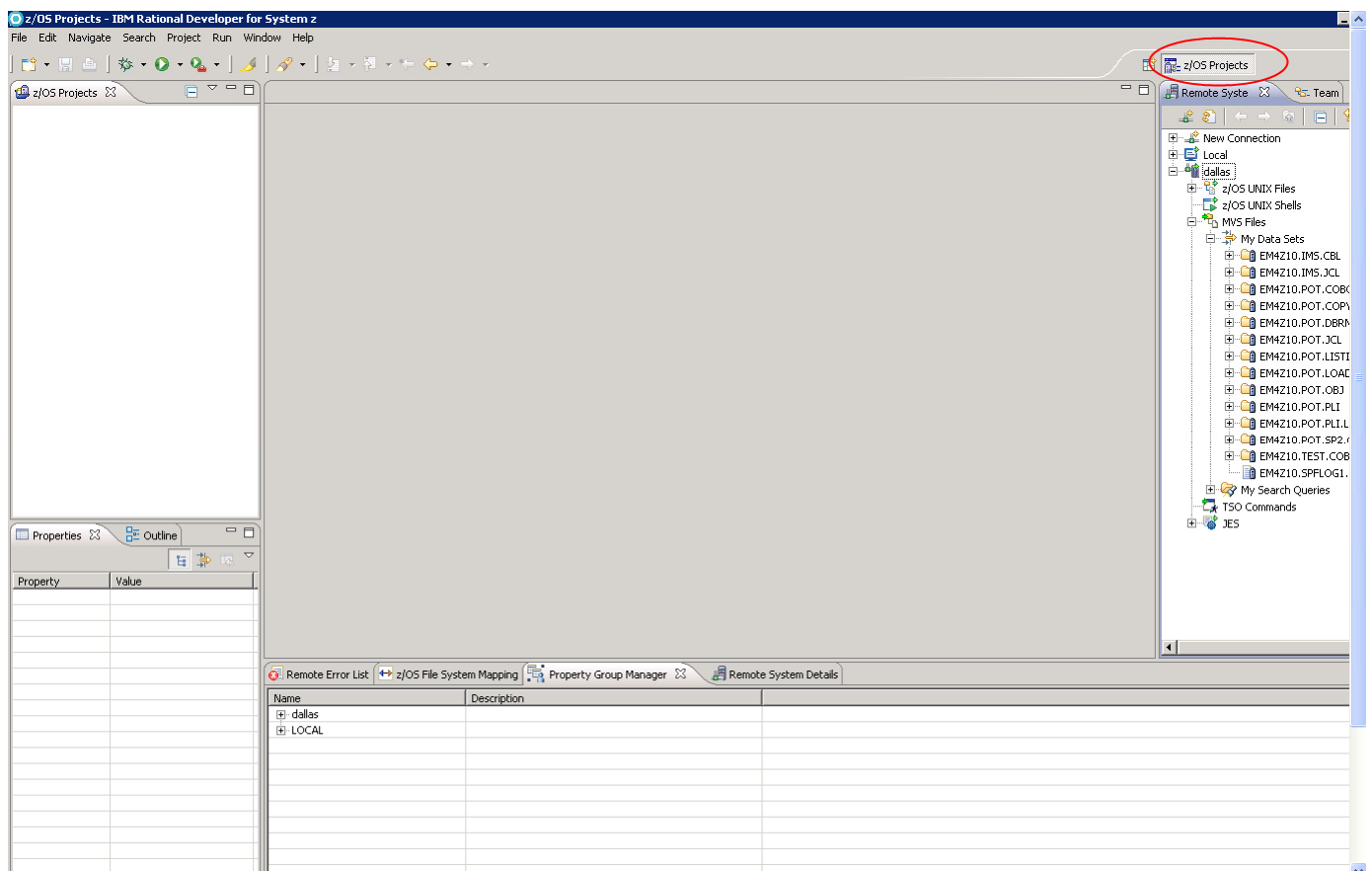
You will now define a remote system and connect to it.

This section is similar to LOGON to a TSO using a provided userid and password.

Before starting this Lab be sure that you have a unique z/OS userid and password. You also could use this userid/password to logon to a z/OS TSO session. If you do not have a username and password you can obtain one from the lab moderator.

1.1 Defining a z/OS Remote system

1.1.1 After you logged on into the sandbox image wait a few minutes for RDz to come all the way up. Once RDz is up you will see an image similar to the one in the figure bellow. Notice that the RDz perspective is selected for you.



What is the z/OS Projects perspective?

Use the z/OS Projects perspective to define, connect, and work with remote systems, as well as create, edit, and build projects, subprojects, and files on your remote systems.

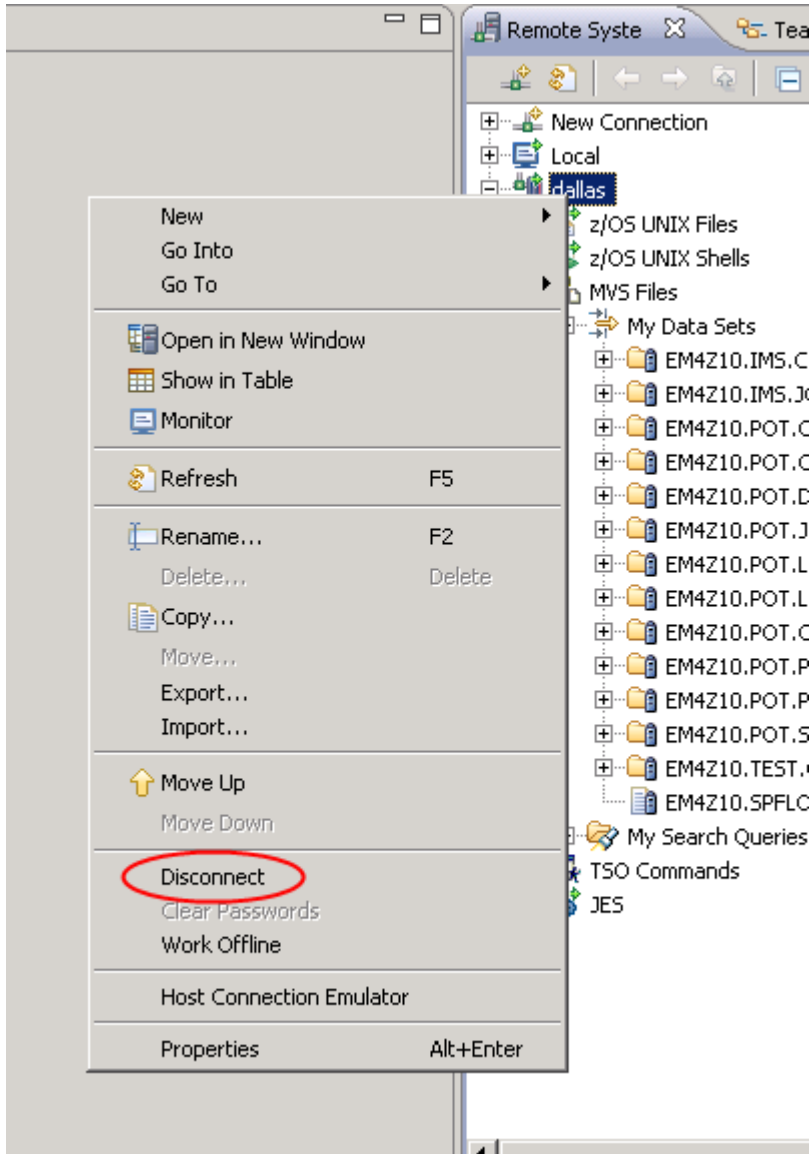


The z/OS Projects perspective contains the following views:
Remote Systems view, z/OS Projects view, Properties view, Outline view, Remote Error List view, z/OS File System Mapping view and Remote System Details view

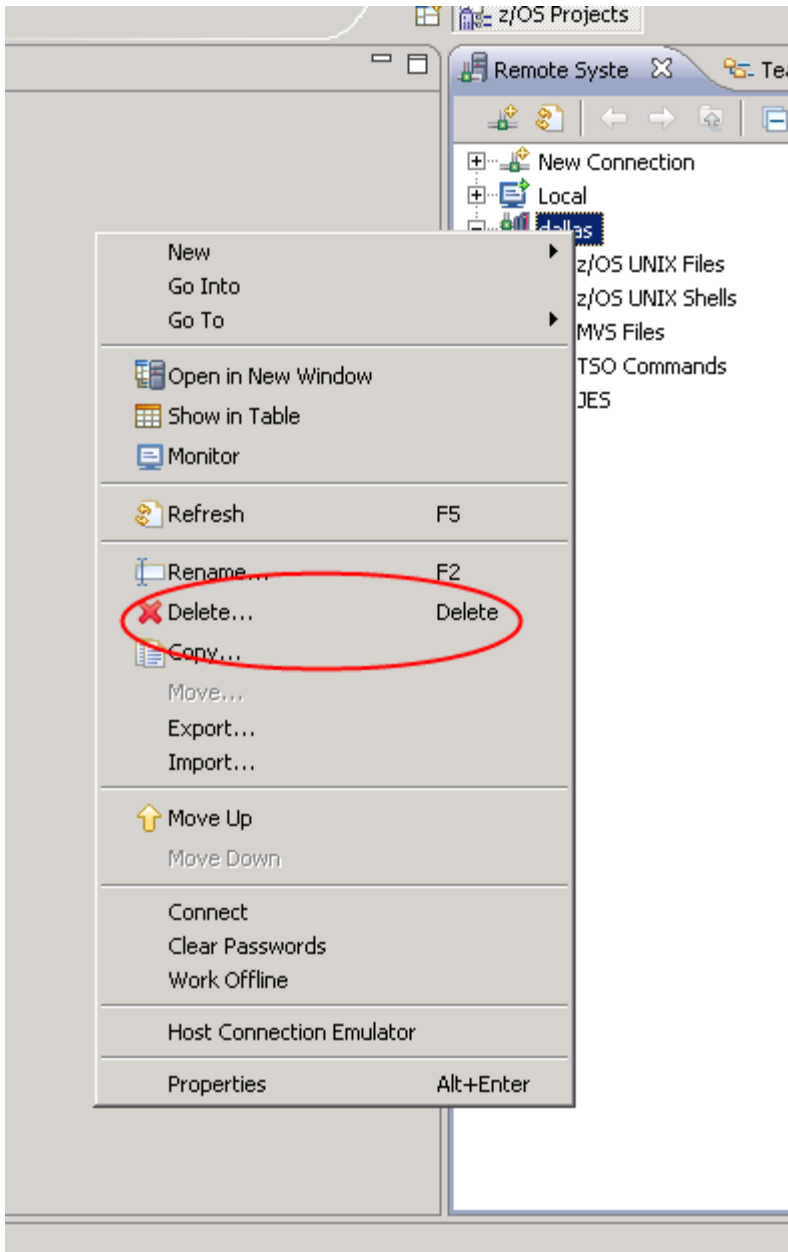
1.1.2 Delete any previous z/OS connection. You will be creating your own in the next step

▶▶ Click on the tab **Remote Systems** view on your right.

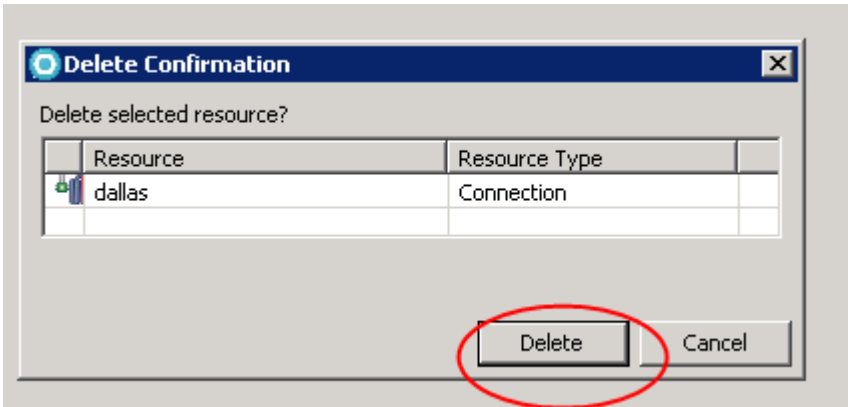
▶▶ Right-click on dallas and then click disconnect.



▶▶ Right-click on dallas again and click delete



▶▶ Click delete

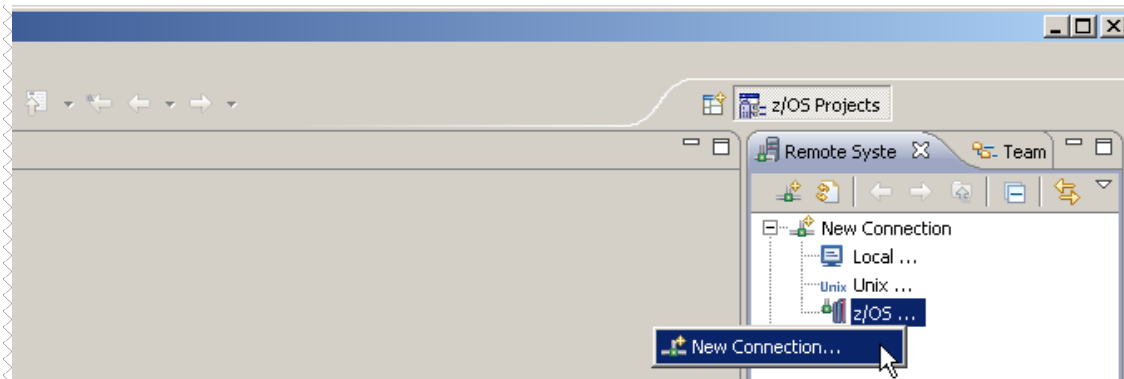


1.1.3 To create your own z/OS connection

▶▶ Click on the tab **Remote Systems** view on your right.

▶▶ In the *Remote Systems* view. Expand the **New Connection** if needed.

▶▶ From the New Connection tree, right-click on **z/OS...** and select **New Connection** to open the pop-up menu.



1.1.4 ▶▶ In the *Host name* field, type **zserveros.demos.ibm.com** as hostname. This is the z/OS machine name. It could be also its IP address instead. (192.84.47.60)

▶▶ In the *Connection name* field, type **dallas**

This label that you assign to this connection will help you to differentiate between multiple connections to the same type of remote system.

▶▶ To verify that the hostname or IP address is valid, select the **Verify host name** check box.

▶▶ Click **Next** to proceed to the JES subsystem properties page

New Connection
Remote z/OS System Connection
Define connection information

Parent profile: demovm1

Host name: zserveros.demos.ibm.com

Connection name: dallas

Description:

Verify host name

< Back Next > Finish Cancel

1.1.5 ▶▶ In the *z/OS Unix Files* be sure that **Remote daemon** is selected, accept the port default and click **Next**

New Connection
z/OS UNIX Files
Define subsystem information

Indicate how the remote server should be launched by default

Remote daemon
Daemon Port (1-65535) 4035

REXEC
Path to installed server on host: dstore
Server launch command: ./server.zseries Port (1-65535): 512
 Auto-detect SSL
 Use SSL for network communications

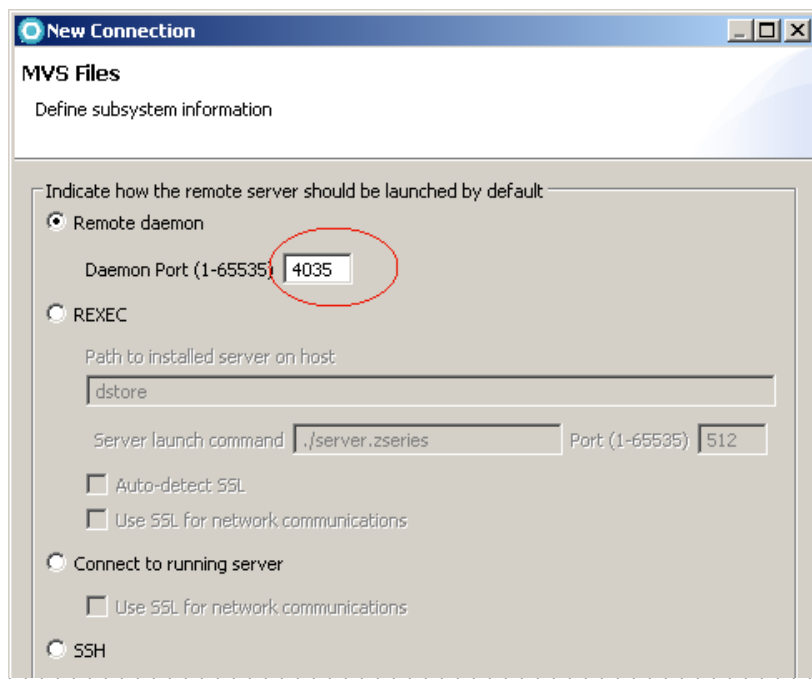
Connect to running server
 Use SSL for network communications

SSH
Path to installed server on host: dstore
Server launch command: ./server.zseries Port: 22
 Password authentication
 Key authentication

< Back Next > Finish Cancel

1.1.6 Note that we could use other ways to connect, besides remote daemon. We will accept all the defaults here..

▶▶ In the *MVS Files* window, be sure that **Remote daemon** is selected, accept the port default and click **Next**

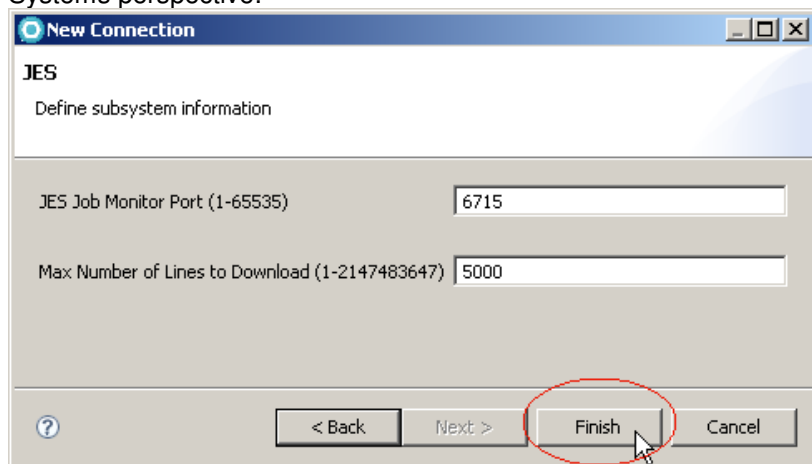


About the screen shots in this paper...

Note that some pictures in this tutorial (like the above) do not show all the buttons. This is because we want to conserve space. The pictures (screenshots) are designed to help you better understand what you are doing..

1.1.7 In the JES Job Monitor Port field, accept **6715** (default) as the port on which the Remote Job Monitor is listening. In the *Max Number of Lines to Download* field, you could 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**.

▶▶ Be sure that the port is **6715** and click **Finish** to create the new z/OS connection and add it to the Remote Systems perspective:

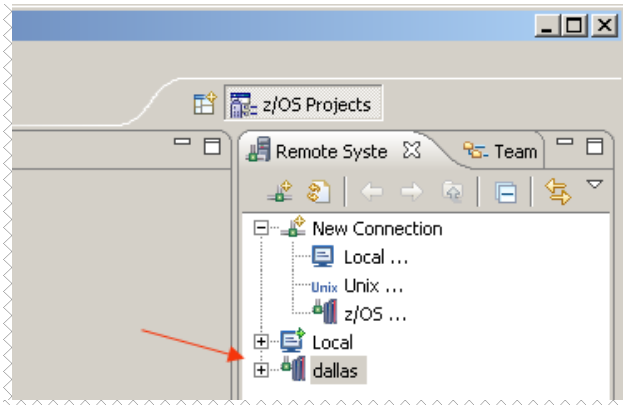







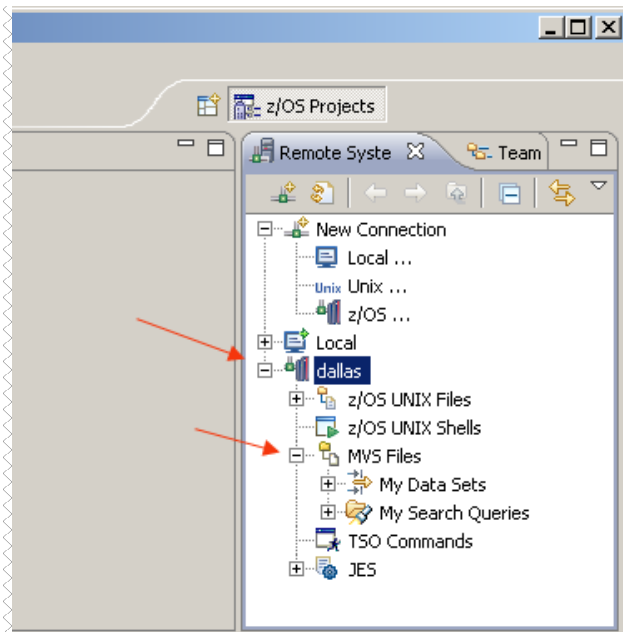
In case of errors...

If you have errors during the connection creation it is because the z/OS system name is not correct or not available (if you specified Verify host name in the step 1.1.5 above). Be sure that you did not type a wrong z/OS name (zserveros.demos.ibm.com). If you still have errors could be because Dallas system is down or you have network issues. Contact the instructor.

1.1.8 If the network is available, you will have the connection **dallas** created as show below:



1.1.9  To better view what's inside the connections expand the folder **dallas**, left-clicking on the  **dallas** and  MVS Files



What have you done so far?

You used the Remote Systems view to define and connect to a z/OS remote system via TCP/IP.

For each system to which you have established a connection, the Remote Systems view shows six main nodes under the connection name: z/OS UNIX files, z/OS UNIX Shells, MVS files, TSO commands and JES.

From the Remote Systems view, you can complete the following tasks:



- Emulate a z/OS session
- Add or remove remote system definitions
- Connect or disconnect remote systems
- Allocate partitioned data sets or sequential data sets on remote systems
- Launch an edit session for a specific file or PDS member
- Migrate, delete, or rename data sets
- Create PDS members within data sets
- Move, copy, delete, or rename PDS members
- Submit jobs to the remote system
- Edit data set name levels
- Add, modify, or remove mappings that associate workstation files with remote files, etc...

Note that a filter named *My Data Sets* is automatically created for you.

This means for example that if you are logged on as *EMPOTXX*, you will see all data sets that will start with *EMPOTXX.** in other words, EMPOTXX would be the High Level Qualifier (HLQ), similar to when you use TSO.

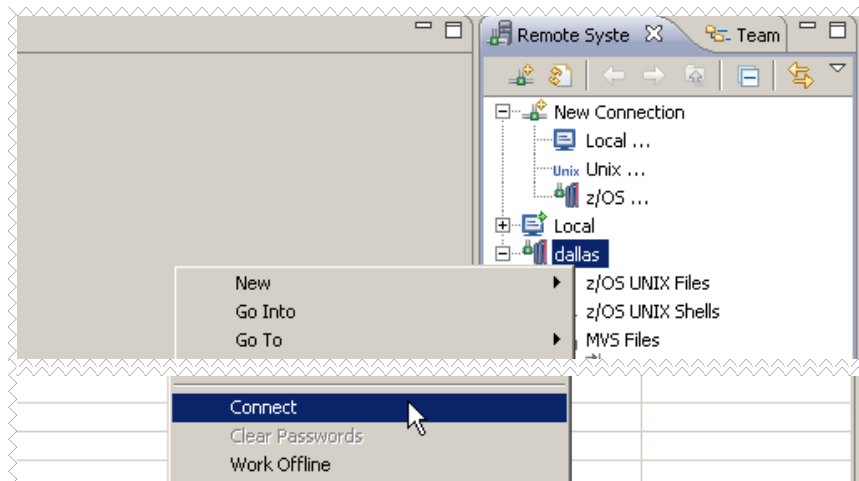
Note also that you could create other filters...

Examples of MVS Files filter are: HLQ.*, HLQ.*.COBOL, HLQ.UTIL.*, HLQ.*.COB*, etc... However, a filter beginning with an * is not a legal filter.

1.2 Connecting to the z/OS Remote system

1.2.1 To Connect to the z/OS system located in Texas:

▶▶ Right-click on **dallas** and select **Connect**:



1.2.2 You will be are prompted for your z/OS userid and password.

▶▶ Type the **assigned userid** and **password**. In the figure shown below we used *EMPOT24*, but your userid will be another one. The password can be any case, don't worry about having it in UPPER case. **Please BE SURE that you are using the right ID and password that were assigned for you.**

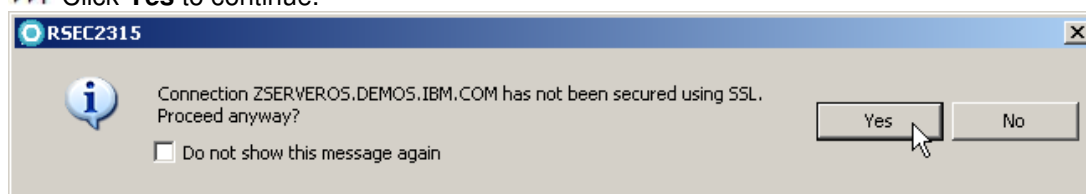
▶▶ Select **Save user ID**. You will be prompted for the password if you disconnect and connect again later.

▶▶ Click **OK** to connect to MVS Files subsystem.

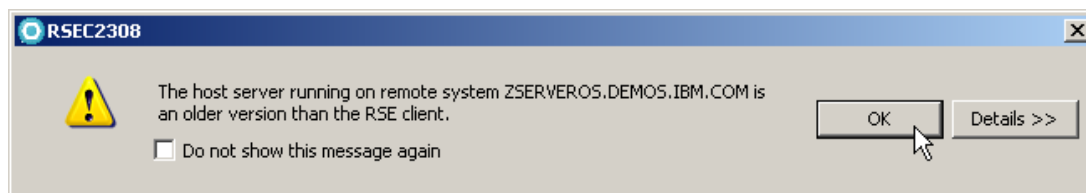


1.2.3 Since our connection is not secured, a message will prompt.

▶▶ Click **Yes** to continue.




1.2.4 ▶▶ Depending when you are doing this lab it may be possible that our z/OS system is using a previous version of RSE. If that is the case the message below may be displayed and you must click **OK**.



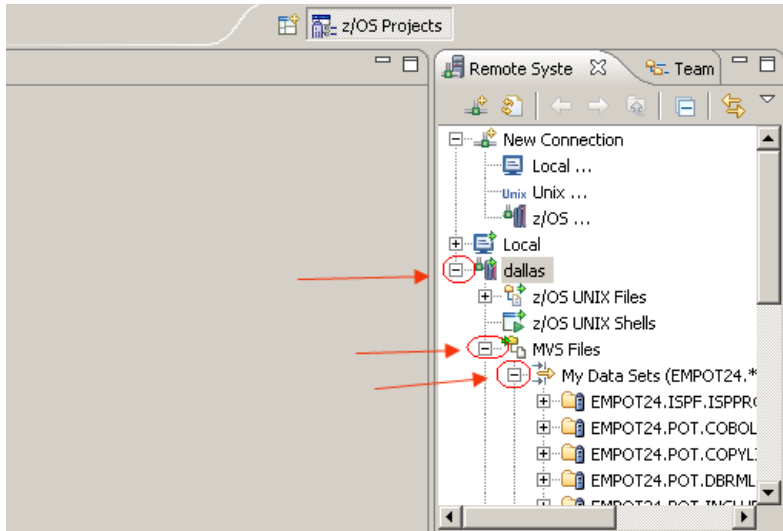
1.2.4 **Be Patient!** The connection could take a while depending on the network conditions. Also remember the overhead that is caused by VMware. In a normal network this connection is very fast. You will be able to see in the bottom of the window a message that shows that the connection is in progress:



1.2.5 If you successfully connect to the remote system, the *dallas* icon changes to 

Expand  MVS Files and  My Data Sets to see all your MVS data sets

Note that you do not have the same PDS's shown below (EMPOT24.*) this is just an example. Depending on which ID you are using you may have no data sets, the ID;s are reused and we have no control what is there.



Remote Systems view

The Remote Systems view shows all existing connections to remote systems. Connections 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.




Section 2 – Allocate z/OS Data sets

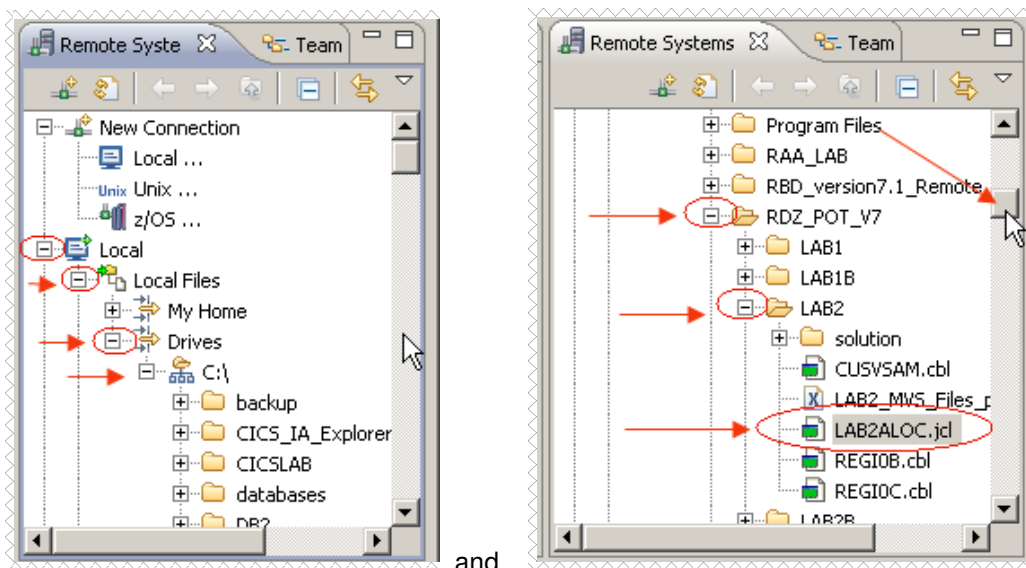
You are connected to a z/OS remote system. Now you will modify a provided JCL and submit it to z/OS to allocate some datasets that are required for this Lab.

Note that the JCL provided is on your local workstation, NOT in the remote z/OS system. You will modify this JCL and submit it for execution in the z/OS remote system.

2.1 Modifying the provided JCL

We provided a simple JCL that will allocate data sets with your assigned z/OS ID, you will edit the provided JCL and change EMPOTXX to your assigned ID where you find it.

2..1.1 Using the *Remote Systems* view left click in the + sign of **Local** (NOT the 'Local' under New Connection), **Local Files**, and **Drives** and  C:\ and look for the file **LAB2ALOC.jcl** under **RDZ_POT_V7** and **LAB2**

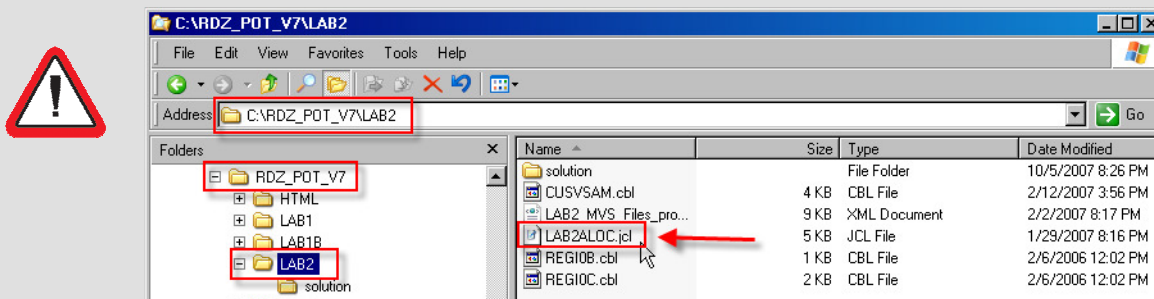


and

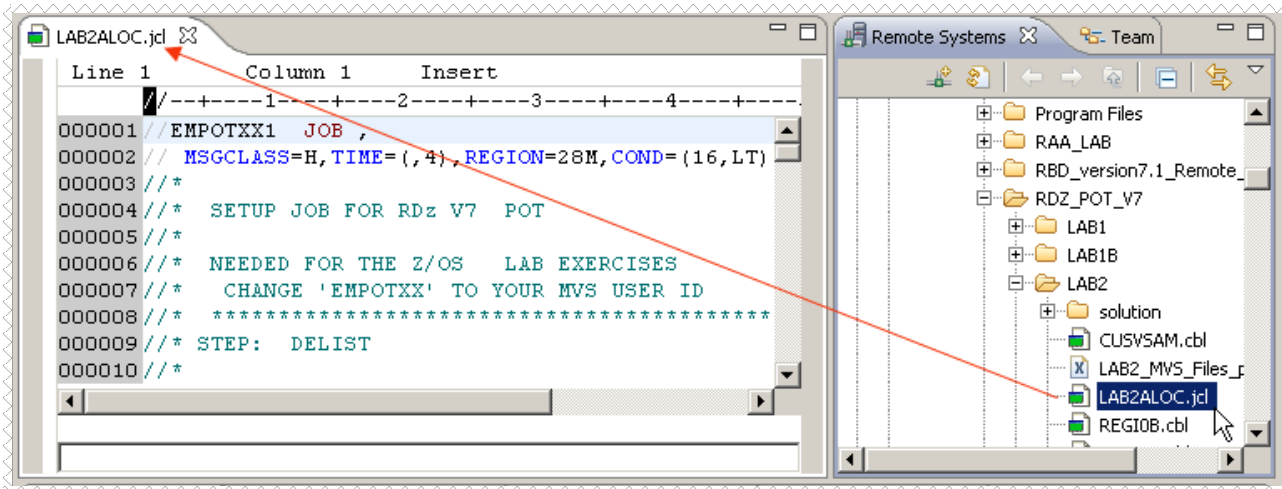
Important!

Be sure that you are selecting the correct file. This JCL file will be used to allocate all the PDS's necessary and copy some members to be used in this Lab.

This file is under the three below... You might have other assets under this folder...



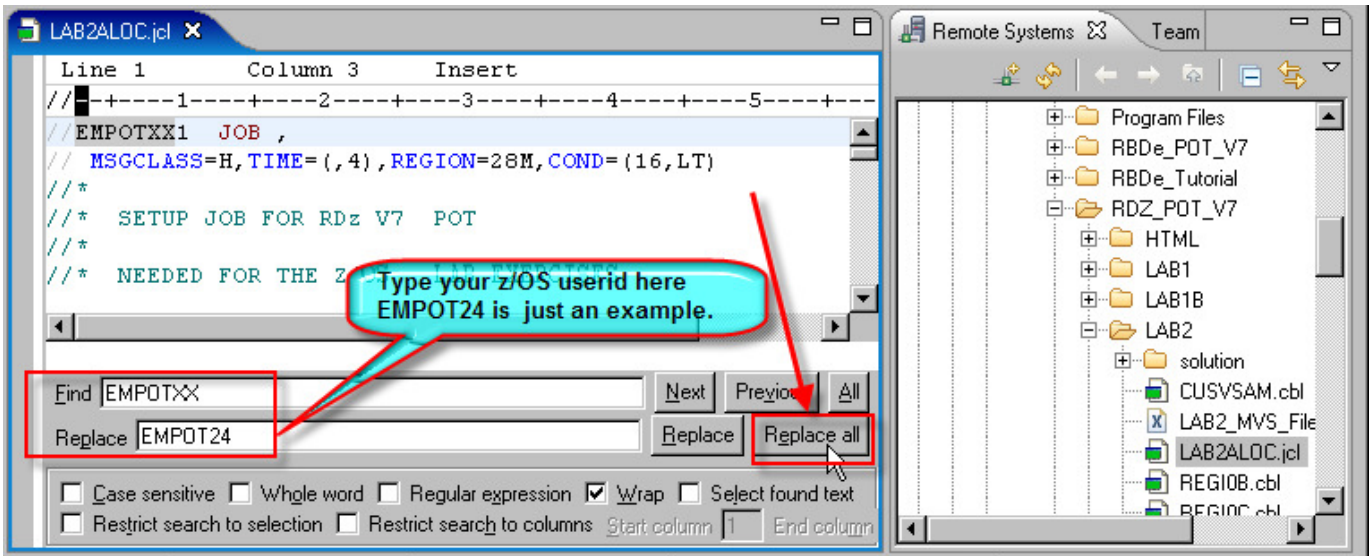
2.1.2 **▶▶** Double click on **LAB2BALOC.jcl** to invoke the LPEX editor



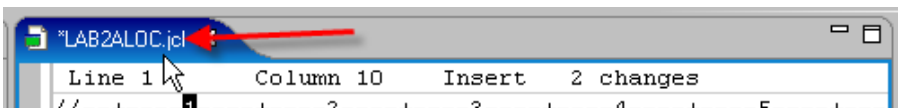
Now you will change **EMPOTXX** to your unique assigned z/OS userid. Your userid should be located after My Data Sets in the Remote Systems tab.

2.1.3 **▶▶** Use **CTRL + F** to find **EMPOTXX**.
Type **EMPOTXX** in the *Find* field and **YOUR Userid** in the *Replace* field and click on **Replace all**.

In this example below we are changing EMPOTXX to EMPOT24.
Be sure that you are using your assigned user ID instead of EMPOT24



2.1.4 **▶▶** **Double click** in the blue title to have a better view of this JCL and be sure that you have changed everything correctly.



2.1.5 **▶▶ Browse** the file to verify the changes.

The idea is to modify the existing EMPOTXX High Level qualifier to your ID .In our example it would be something similar to what you see below. But your change will be different; instead of EMPOT24 you would have your Userid.

```
*LAB2ALOC.jcl
Line 1      Column 10      Insert      2 changes
//-----1-----2-----3-----4-----5-----6-----7---|
EMPOT241 JOB ,
MSGCLASS=H, TIME=(,4), REGION=28M, COND=(16,LT)
/*
/*  SETUP JOB FOR RDz V7  POT
/*
/*  NEEDED FOR THE Z/OS  LAB EXERCISES
/*  CHANGE 'EMPOT24' TO YOUR MVS USER ID
/*  *****
/* STEP:  DELIST
/*
/*  DE-ALLOCATE DATASETS NEEDED FOR POT
/*  (PRIOR TO ALLOCATING THEM IN THE NEXT STEP)
/*  *****
//DELIST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
DELETE EMPOT24.POT.COBOL
IF LASTCC = 8 THEN SET MAXCC = 4
DELETE EMPOT24.POT.COPYLIB
IF LASTCC = 8 THEN SET MAXCC = 4
DELETE EMPOT24.POT.OBJ
IF LASTCC = 8 THEN SET MAXCC = 4
DELETE EMPOT24.POT.PLI
IF LASTCC = 8 THEN SET MAXCC = 4
DELETE EMPOT24.POT.LISTING
IF LASTCC = 8 THEN SET MAXCC = 4
DELETE EMPOT24.POT.PLI.LISTING
IF LASTCC = 8 THEN SET MAXCC = 4

Find EMPOTXX
Replace EMPOT24
```

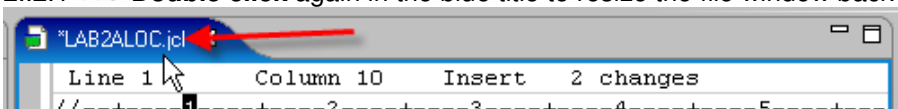
2.1.6 **▶▶ Save** the changes using **Ctrl + S**. Note that the * that is next to the title goes away:



2.1.7 **▶▶ Press Esc** key to move the cursor back to the command line at the bottom of the editor.

2.2 Submitting a JCL to z/OS execution

2.2.1 **▶▶ Double click** again in the blue title to resize the file window back to its original size.



2.2.2 If you are still connected to the JES subsystem, you can submit this job for execution to *dallas*.(sometimes a JES timeout occurs. If that is the case you will have an error when submitting and we will explain how to fix it)

▶▶ Type **submit to dallas** at the command line and presses **enter**

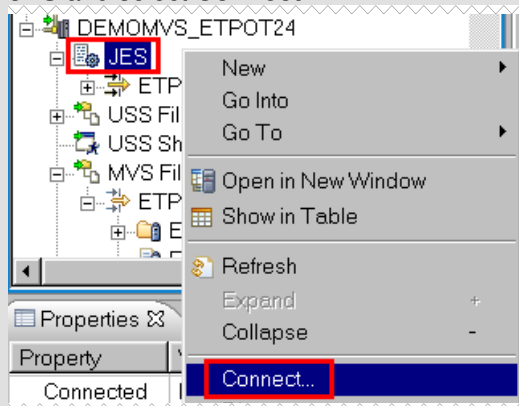
```
000009/** STEP: DELIST
000010/**
submit to dallas
```

If you have an error message (could be due to a time out), execute the steps below, otherwise proceed to step 2.2.3

If the Job is not submitted and you get the message *Job Monitor not connected to system as* shown below, Could be because you are not connected to the JES subsystem.

```
000041/** STEP: ALLOC
Job Monitor not connected to system: demomvs
submit to demomvs
```

▶▶ Using the Remote Systems view, locate the node JES under **dallas** and **Right-click** on **JES** and select **Connect**.

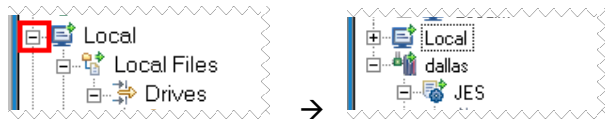


▶▶ Once you have connected, execute the step 2.2.2 above. If you cannot connect contact the instructor. The Job Monitor was not started in the z/OS system

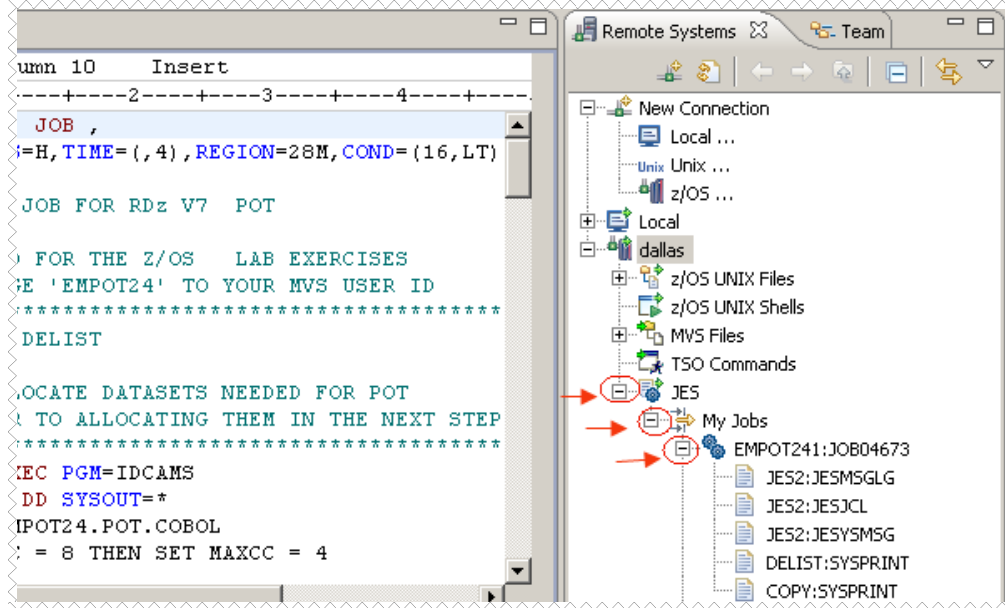
2.2.3 You will see the JOB ID that was created for this execution

```
JOBID: JOB00586
```

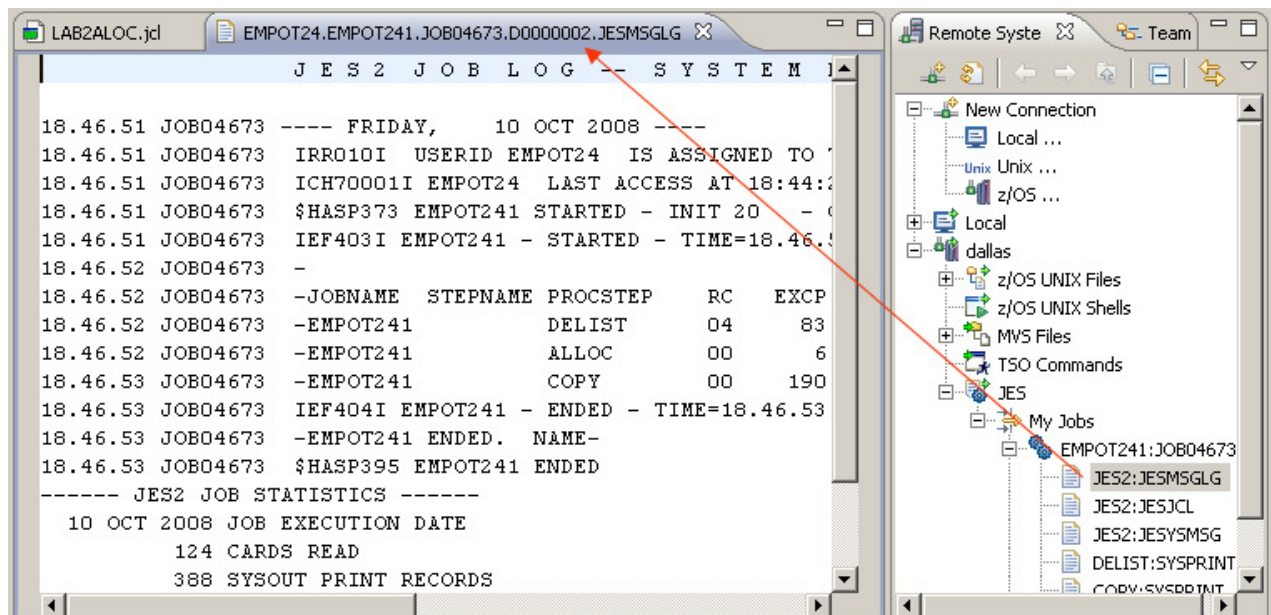
2.2.4 **▶▶** Using the *Remote Systems View*, collapse the local folder by left clicking on the “-” sign, since we don’t need that anymore.



2.2.5 **▶▶** Using the Remote Systems view expand the **JES** node, then **My Jobs** filter and **EMPOTXX1** (left click on + or your userid).



2.2.6 **▶▶** **Double click** in the JES Job Log (**JES2:JESMSG LG**) and be sure that the job has successfully executed and has the return code is 4, 0 and 0 as shown below: (note that it could be all zeros if you had datasets allocated already). The DELIST step might have 4 or 0. If the dataset was already allocated it will delete it and in this case will be 0 instead of 4.



What have you done so far?

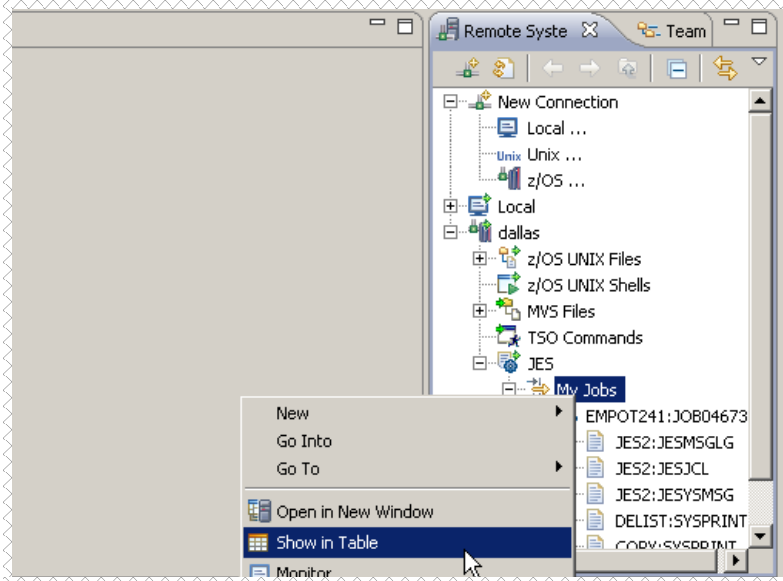


You just submitted a JOB that was executed in the z/OS and allocated some data sets that you will need for this lab. You saw the execution output above.
Do not continue until this task has been successful.

2.2.7 **▶▶** Close all opened editors: **LAB2BALOC.jcl** and the JES listing by using **CTRL + Shift + F4** or Clicking on the **✕**

2.2.8 Another way (better) to see the JES output is creating another view..

▶▶ Right click on **My Jobs** and select **Show in Table**:



2.2.9 The *Remote System Details* view is opened and some info is also shown. Note the Column that shows the *Return Code*.. If you double click here the output listing will open as well

The screenshot shows the 'Remote System Details' window with a table of job information. A red arrow points to the 'Return Code' column, which contains the value 'U0004' circled in red.

Resource	Job ID	Job Name	Job Owner	Job Entry D...	Return Code	Return Info	System ret...	User re
EMPOT241:JOB04673	JOB04673	EMPOT241	EMPOT24	2008/10/10...	U0004	NORMAL		004

2.2.10 You will now purge the output of jobs listed. If RACF® allows you to do that..

▶▶ To purge Go to the *Remote System* or *Remote System Details* view, **Right-click** on your Job and select **Purge**:

Remote system filter My Jobs

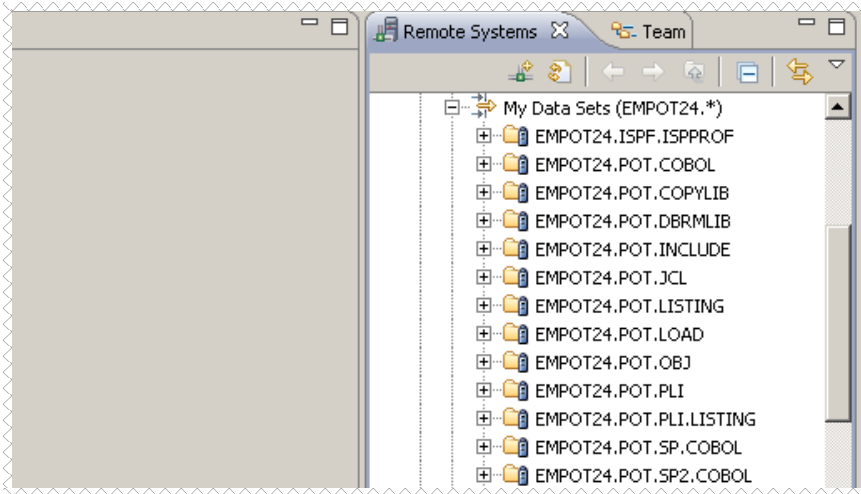
Resource	Job ID	Job Name	Job Owner	Job Entry D...	Return Code	Return Info	System ret...	User re
EMPOT241:JOB04673	JOB04673	EMPOT241			U0004	NORMAL		004
STEP1:JOB01329	JOB01329	STEP1			U0000	NORMAL		000

Context menu options: Go Into, Refresh, Open, Hold, Cancel, Purge

Not authorized for job JOB04673

If you are not authorized: ignore it. RACF® does not allow you to purge.

2.2.11 **▶▶** Using the *Remote Systems view*, expand **My Data Sets** left clicking on the **+** and you should now have all the required datasets for the next exercise allocated. If **My Data Sets** was already expanded from step 1.2.5. Right click on **My Data Sets** and select refresh.



Section 3 – Associate z/OS resources to properties

You have the data sets required on the z/OS and you will define the settings required to work with the z/OS assets, such as specifying the correct COBOL libraries, CICS and DB2 settings, etc... Usually this is done only once using the Properties groups and shared among users.

NEW on Rational Developer for System z version 7.5 – PROPERTY GROUP

You can create a property group with property values that can be shared by z/OS projects, subprojects, and resources.

A property group is a set of property values that you define for local or remote systems. Once defined, the values in a property group can be applied to the z/OS projects, subprojects, and remote resources that you create on that system. Property groups provide a way to manage resource properties, share them easily across systems, projects, resources, and users, and maintain consistency in your development and build environment.



You can, for example, define a property group with values required for debugging in your environment and apply that property group to your resources when you need to debug the programs in your project or subproject. If you need to change a specific property value, for example, the JCL job card and data set, you can change this property once in the property group and the change is propagated to all resources associated with that property group.

System programmers can create property groups and default property values and make them available to users. When a connection is made to a system, Rational Developer for System z searches the system for system property group and default value files. If these files are found, then those property groups or default values are loaded and can be used.

Rational Developer for System z assigns a set of default properties for the set of system properties. If you plan to develop COBOL applications in the workbench, you may wish to set properties on the Compiler Options tab of this page and override the JCL Procedures that would otherwise be used for compile, link, and execution requests.

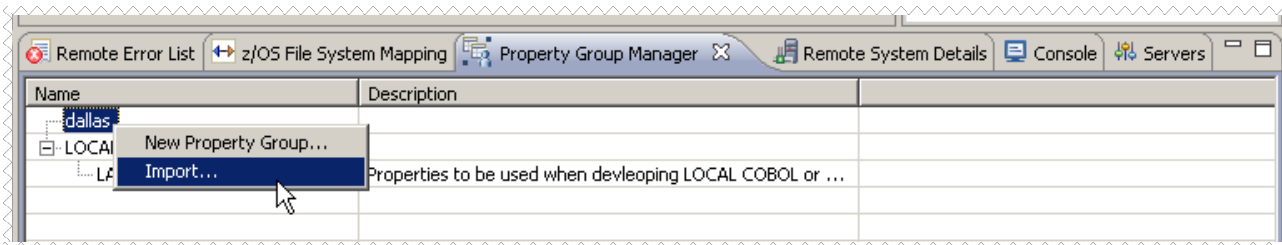
Note that the string <HLQ> is automatically replaced with the high level qualifier you select when you create a project from this system definition. Depending on your use of CICS, DB2, and IMS, you may need to set additional properties on the remaining tabs of this page.

3.1 Associate the property group to the MVS Files

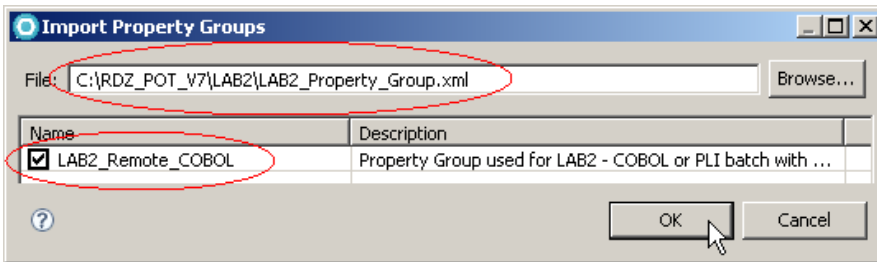
3.1.1 Since this task is done once per installation, we have already created a few property groups on your workspace that reflect our z/OS system used in this exercise.

You still need to import those properties to your workspace.

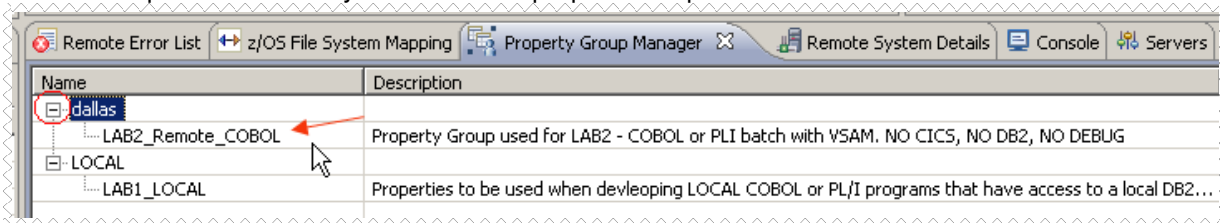
▶▶▶ Using z/OS Projects perspective and Property Group Manager view, right click on **dallas** → **import...** ,




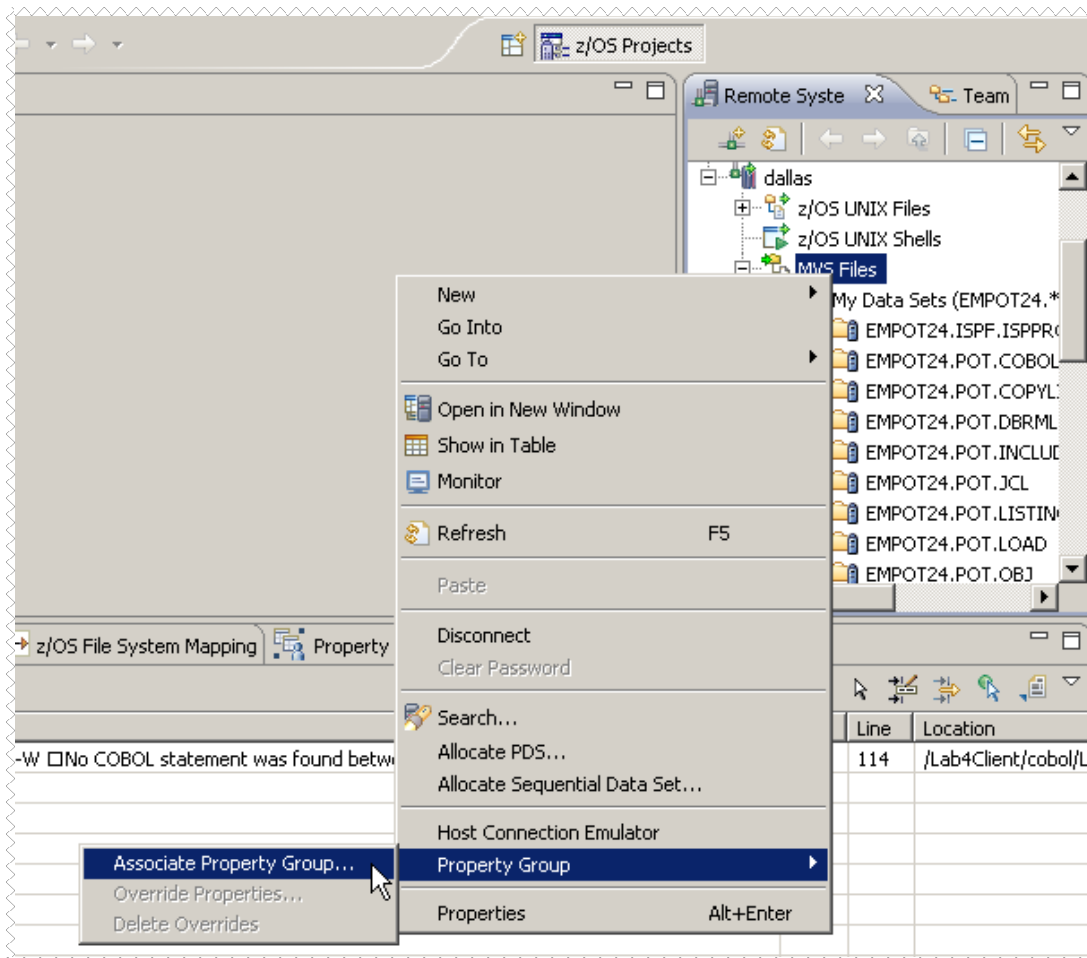
3.1.2 ▶▶▶ Click on **Browse...**, navigate to **C:\RDZ_POT_V7\LAB2\LAB2_Property_Group.xml** select **open** on the *Browse Window* and then select **LAB2_Remote_COBOL** and click **OK**



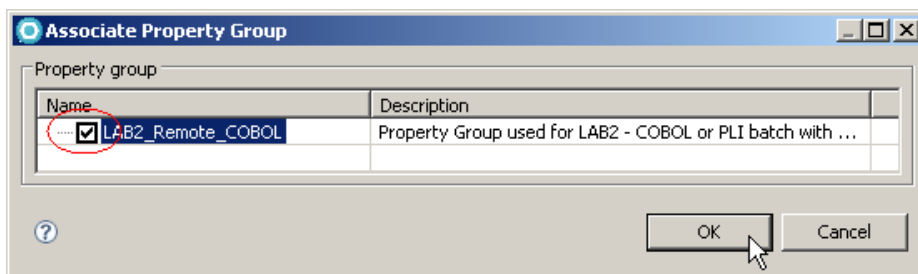
3.1.3 ▶▶▶ Expand **dallas** and you will see the properties imported



3.1.4  Using the *Remote Systems* view, expand **dallas**, select **MVS Files**, right-click and select **Property Group** → **Associate Property Group ...**

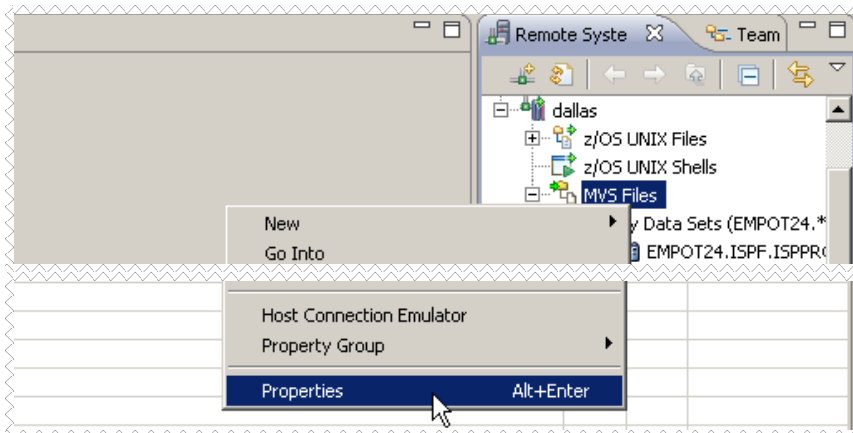


3.1.5  Select **LAB2_Remote_COBOL** and click **OK**

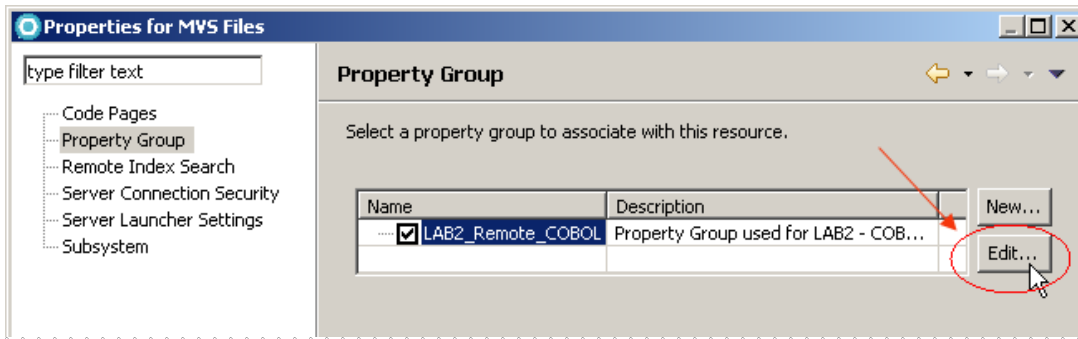


You will see the properties associated to this resource in the next steps. Usually those properties are defined once and shared among developers.

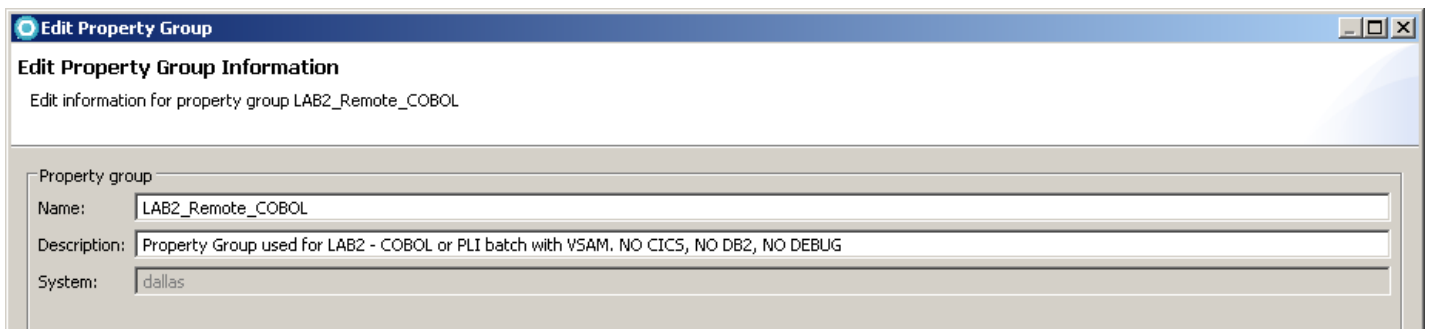
3.1.6 ▶▶ Select **MVS Files** again, Right-click and select **Properties**.



3.1.7 ▶▶ In the Properties dialog, click on **Property Group** and click **Edit**

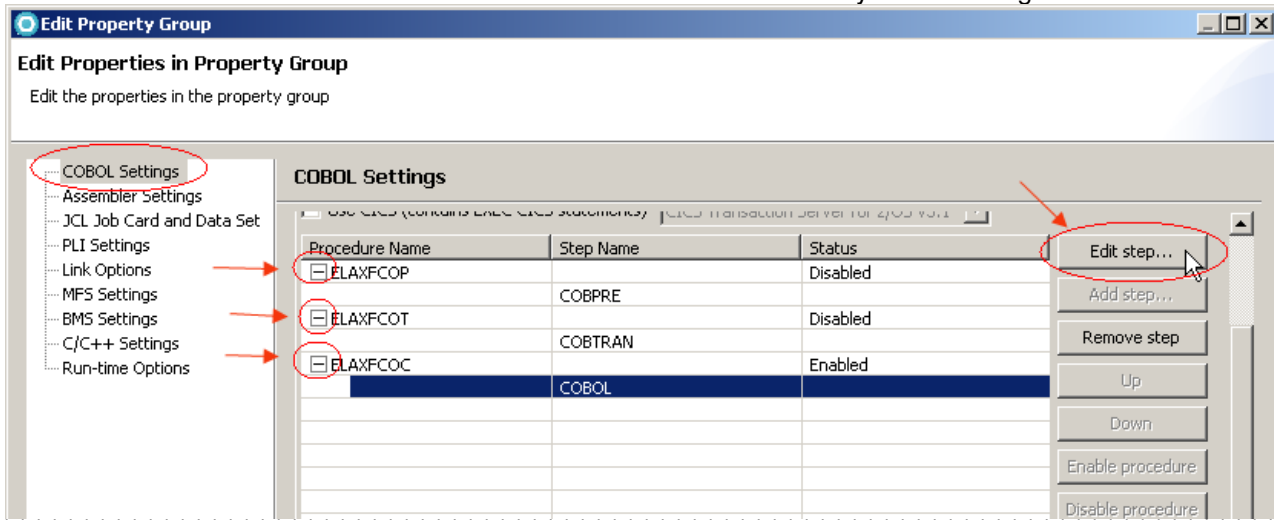


3.1.8 ▶▶ Click **Next** twice from the dialog below



- 3.1.9 ▶▶▶ Using the *Edit Property Group* dialog click on **COBOL Settings** (left).
- ▶▶▶ Click on + sign that is on left of **ELAXFCOP**, **ELAXFCOT** and **ELAXFCOC**
- ▶▶▶ Select **COBOL** and click **Edit step..**

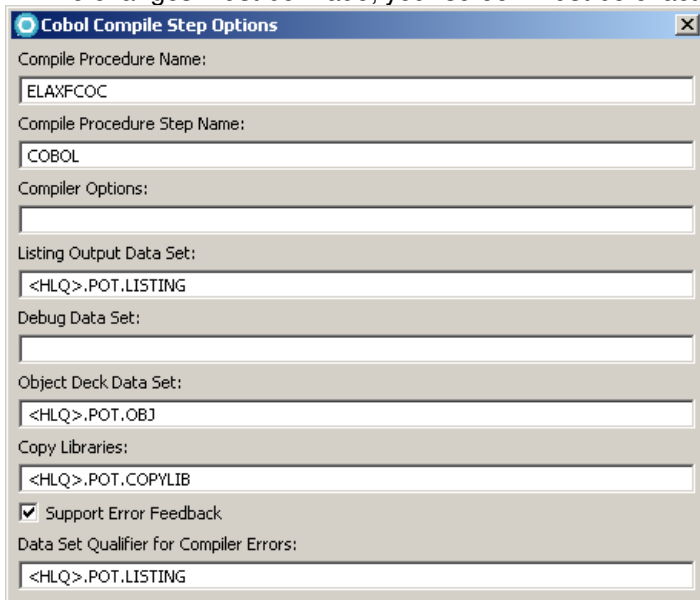
This is the default COBOL Procedure used when the Remote COBOL syntax checking is used.




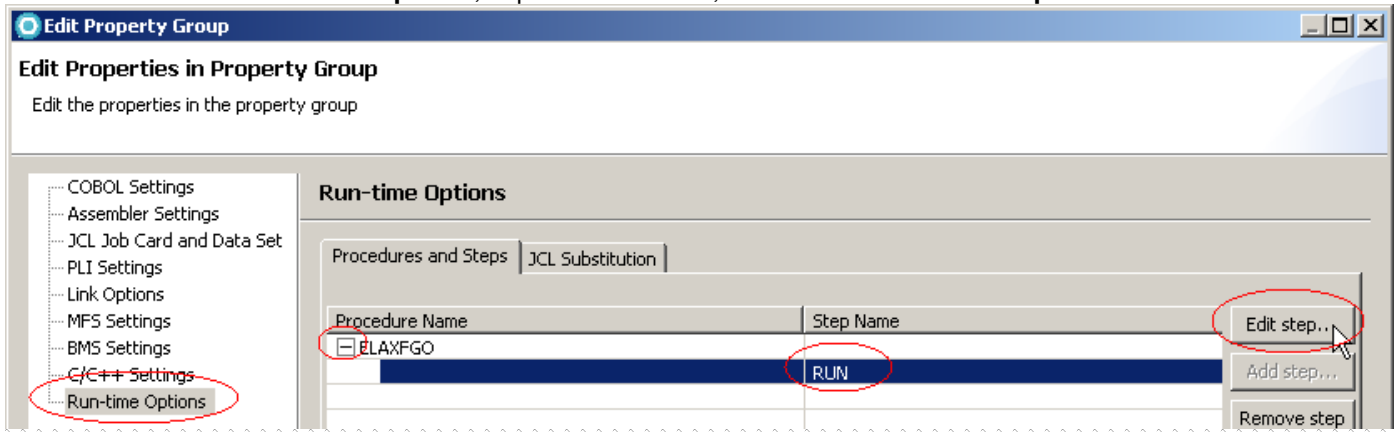
Note that we could enable and disable the procedures that must be executed before the COBOL compilation if necessary. The default supplied procedures (but disabled) are:
 ELAXFCOP – invoke DB2 pre-compiler
 ELAXFCOT – invoke CICS translator
 ELAXFCOC – invoke COBOL compiler
 But by default only ELAXFCOC is enabled. This can be modified if necessary.

3.1.10 The *COBOL Compile Step Options* properties dialog will open. Note that you could change these properties, such as Compile Options, data set names, etc. For instance, when JCL is generated for Compile, it will use the options specified here. The <HLQ> will be replaced with your logon userid when JCL is generated or when you create a z/OS project that we will discuss later.

- ▶▶▶ No changes must be made; your screen must be exactly as shown below. Click **OK** to close it.




3.1.11  Click on **Run-time Options**, expand **ELAXFGO**, click on **Run** and **Edit Step..**

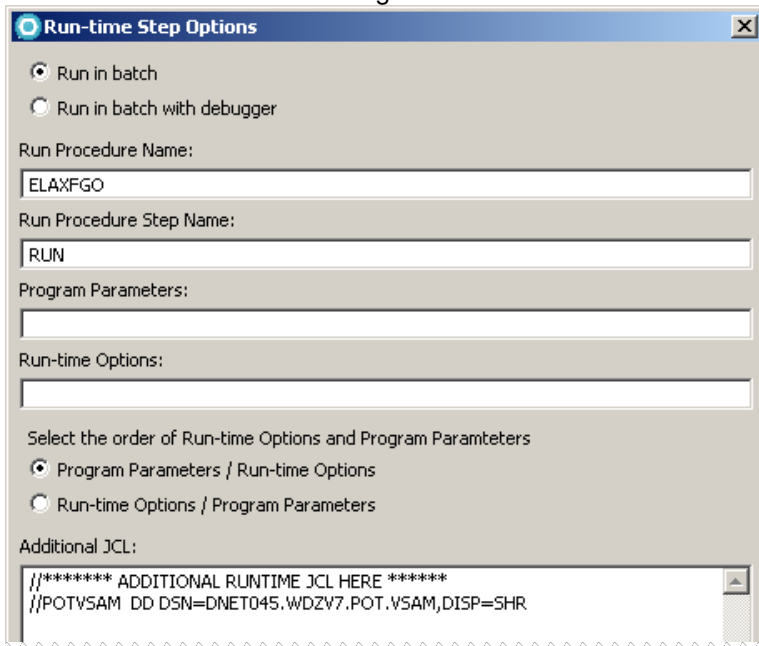



3.1.12 No changes must be made, your screen must be exactly as shown below.

Also note that **Run in batch** was specified. We do not want to use the debugger.

Note the DDNAME POTVSAM in the additional JCL. This will be the DDNAME that will point to the VSAM data set in the z/OS and required for the batch execution. When the JCL is generated this DDNAME will be added..

 Click **OK** to close this dialog.



3.1.13  Click **Finish** to close the dialog and **OK** to close the property Group dialog.

What have you done so far?



You have connected to z/OS and have submitted a Job to allocate and copy some members required in order to do this lab.
You assigned the MVS resources to pre-defined properties .
Now if you ask for JCL generation, for instance, the data set names will be correct..
Now let's copy a COBOL program to z/OS and play with it.

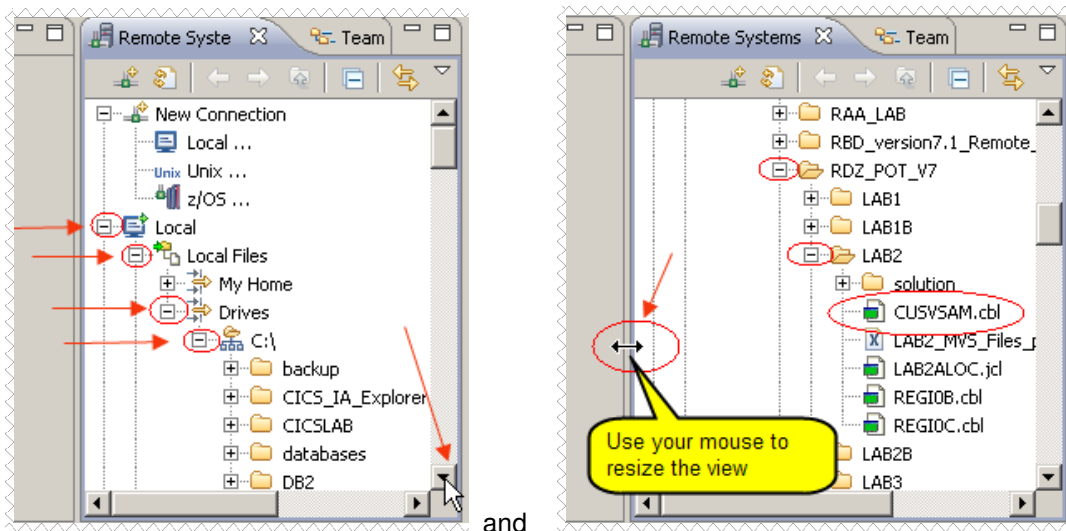
Section 4 – Send the COBOL program to the z/OS

You have the data sets and copy book members required for this lab, but you still need a COBOL program to play with. On this section you will copy a COBOL program from the workstation and move it to your PDS member. Later we will work with this program.

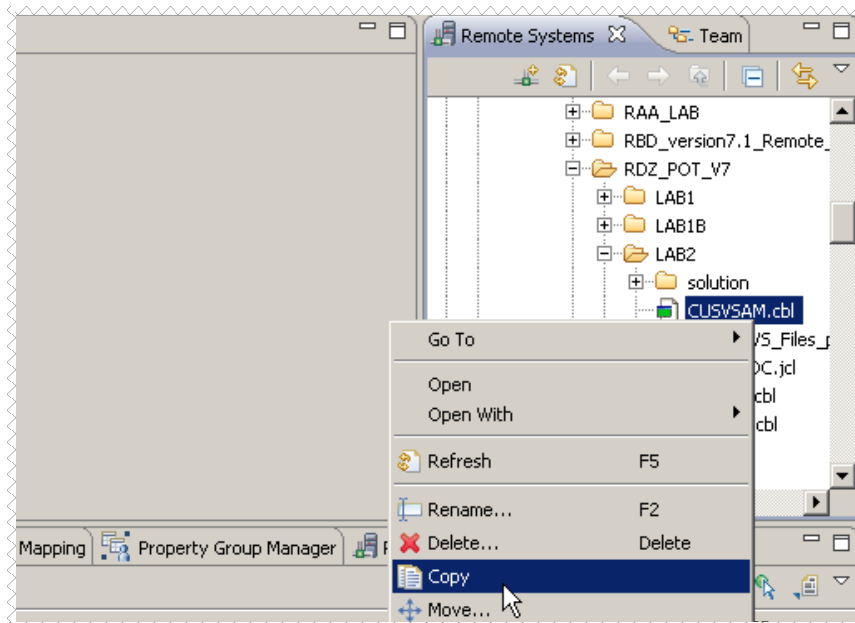
We could do that with the JCL that you had submitted before, but we want to show you that one way to move data sets is to copy/paste files from your workstation to the z/OS. This capability will also work between different z/OS connected systems.

4.1 The data sets that you created submitting the JCL consist solely of the COBOL Copybooks and two COBOL subroutines. You will now copy the main COBOL program that is in your windows directory to the z/OS system.

▶▶ Using the Remote Systems view left click in the + sign of the nodes **Local**, **Local Files**, **Drives** and **C:** and look for the folder **C:\RDZ_POT_V7\LAB2** in the next step you will copy **CUSVSAM.cbl** to the PDS allocated on z/OS

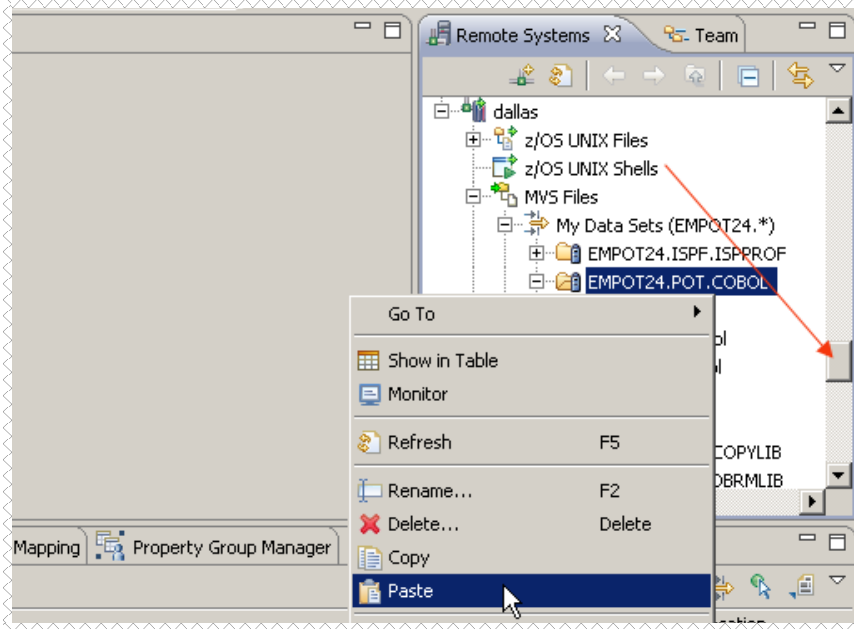


4.2 ▶▶ From *Remote Systems* view right click on **CUSVSAM.cbl** and select **Copy**

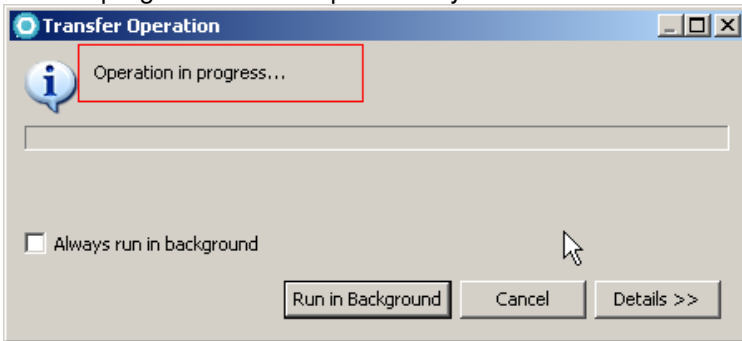


4.3 Still using the *Remote Systems* view, scroll down to the **dallas** connection and if not already expanded, expand **MVS Files** and **My Data Sets** until you find the dataset **EMPOTxx.POT.COBO**. (where EMPOTxx is your userid) right-click on **EMPOTxx.POT.COBO** and select **Paste**.

Note: If you do not see the EMPOTxx.POT.COBO, right click on **My Data Sets** and click **Refresh**

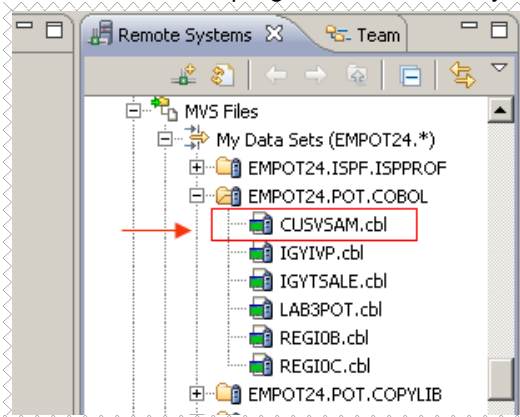




4.4 The programs will be copied from your workstation to the z/OS. A progress indicator will be shown:

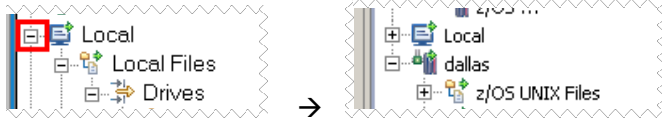


4.5 Expand the dataset **EMPOTXX.POT.COBO** left clicking on **EMPOT24.POT.COBO** and you will see the COBOL program (**CUSVAM.cbl**) loaded at the z/OS and also mapped to cbl (COBOL).

Note that the other programs were already there and were copied when you submitted the JCL on section 2.



4.6  Using the *Remote Systems View*, scroll back and collapse the local folder left clicking on the  Local , since we don't need it anymore.



What have you done so far?



You have connected to z/OS and have submitted a Job to allocate and copy some members required to do this lab. You also associated some properties to the MVS files that you will work with.

In this last section you copied a COBOL program from your local workstation to a z/OS dataset. You are ready to start working on this COBOL program now..

Section 5 – Create a z/OS Project and MVS Subproject

To make your job easier, you will group together all the assets that you will work with. This is a new development concept for TSO users, since TSO does not provide such capability. To accomplish this task you will create a z/OS project and select which assets we will be part of this project.

What is a z/OS project?

After you define a z/OS-based system, you can define a z/OS project under that system. You can define the z/OS project, however, only when you are connected to the system. Rational Developer for System z assigns a set of default properties from the set of system properties. However, changes that you make to system properties do not affect the definition of an existing project. If you change your system properties to reference a new compiler release, for example, the reference affects only those projects that are defined subsequent to the change. This isolation of system data from existing projects is beneficial because it lets you develop your code without disruption. You can introduce changes to the project definition at a time of your choosing.

States of a z/OS project



A z/OS project is in either of two states:

- In online state, the project is connected to the system to which the project refers. You can directly change the data sets that are stored in that system.
- In offline state, the project can access only workstation-based files, which may be new or may be copies of mainframe resources.

When you disconnect from z/OS, you can specify the data sets and members to be transferred to the workstation. When you switch back to the online state, the specified files are automatically uploaded to the mainframe, with a confirmation message that keeps you from unintentionally overwriting resources.

Creating a new MVS subproject

MVS subprojects contain the development resources that reside on an MVS system. You can create multiple subprojects in a z/OS project.

Before you create an MVS subproject, you need to have completed the following tasks:

- Connecting to a remote system
- Creating a z/OS project

What is new on Rational Developer for System z Version 7 related to z/OS Projects?

Host-based projects: A host-based project is one that has been defined on a z/OS system and can be downloaded to the workstation when you connect to the remote system. Host-based projects enable an installation to define and automatically propagate projects on client workspaces from a central location.



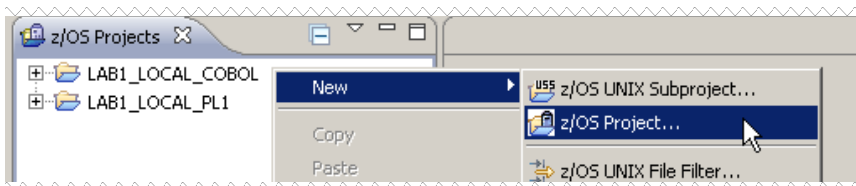
When you disconnect from a remote system, the host-based projects are removed from your z/OS Projects view.

Host-based projects are downloaded automatically when you connect to a remote system. When they are defined on the z/OS system, host-based projects are associated with specific user IDs and downloaded when those user IDs connect to the z/OS system.

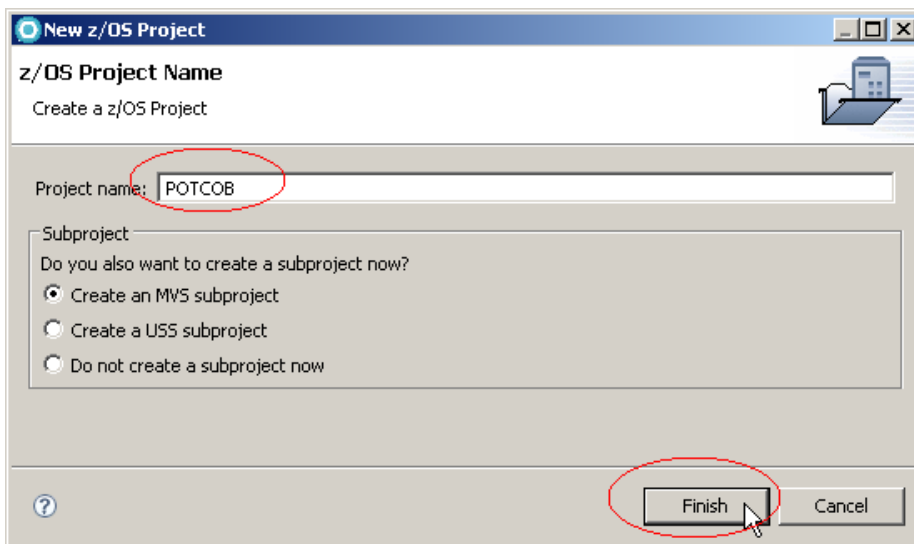
5.1 Creating a new z/OS Project.

The advantage of creating a z/OS Project is that we just focus on those datasets and members that are being constructed or updated, instead of having all of the dozens of mainframe datasets or members. At anytime if you need to see a dataset not added to the project, just go to the z/OS Systems view and add it. Also, at any time, you can remove from your project the datasets no longer being used.

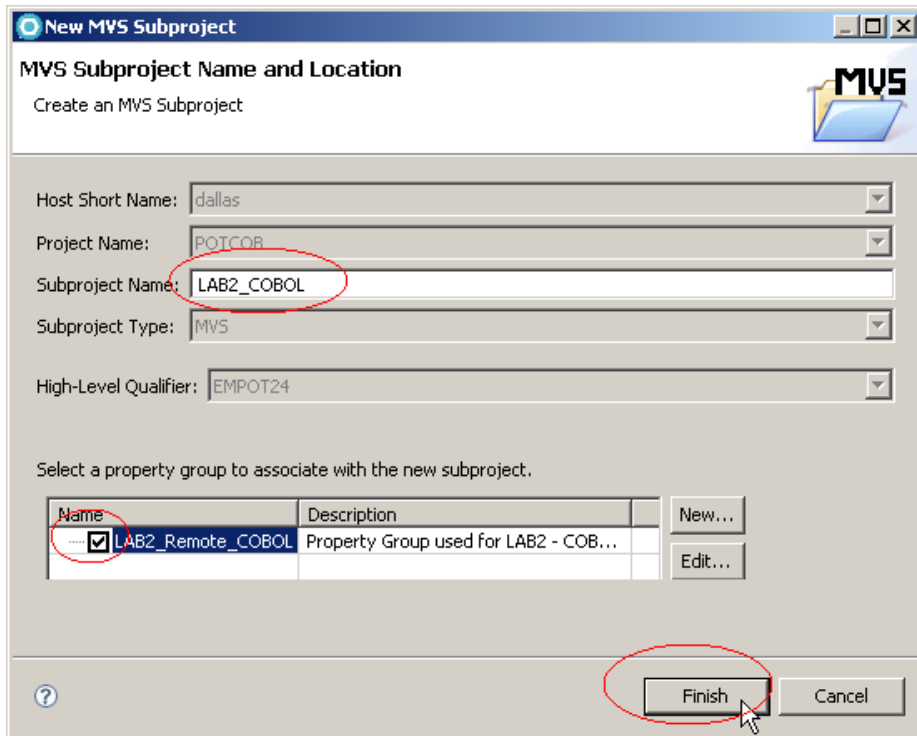
5.1.1 ▶▶ Using the z/OS Projects view (on the left), right click the blank area and select **New → z/OS Project...**



5.1.2 ▶▶ Type **POTCOB** as the *Project name* select **Create an MVS subproject** and click **Finish**.

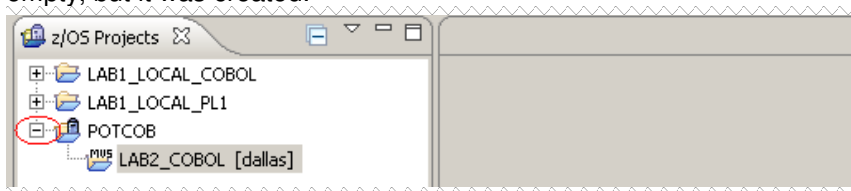


5.1.3 ▶▶ Type **LAB2_COBOL** as the *Subproject Name* click on **LAB2_Remote_COBOL** property group and click **Finish**



You should see a z/OS Project named *POTCOB* in your z/OS Projects view.

5.1.4 ▶▶ Using the z/OS projects view, left click on the + sign to expand the project **POTCOB**, you'll see that is empty, but it was created.

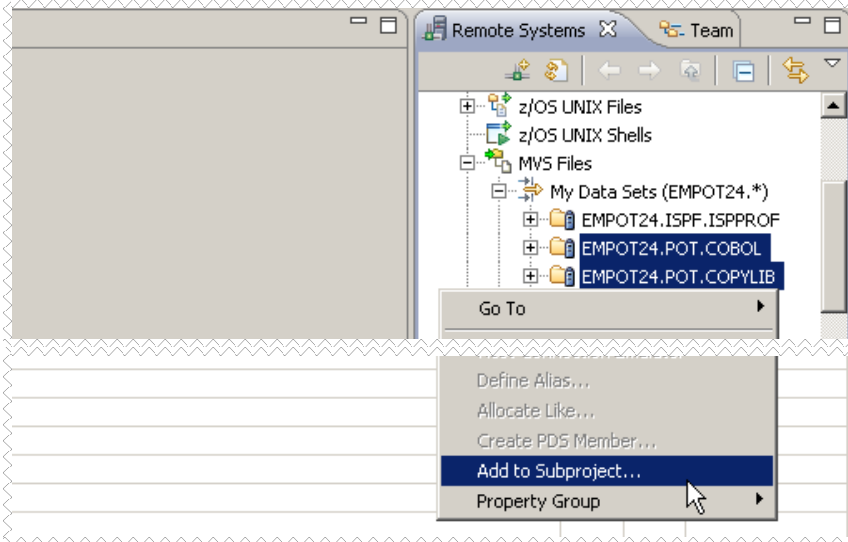


5.2 Add resources to the subproject

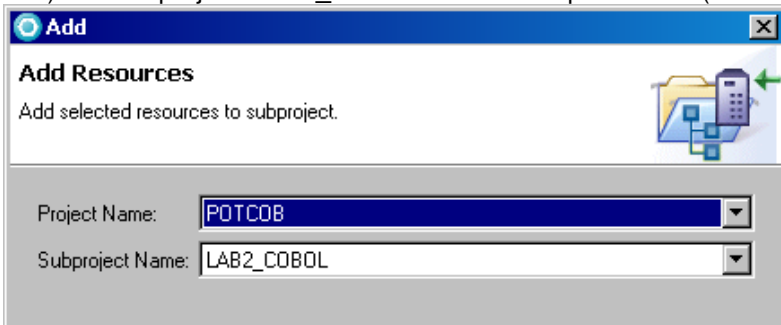
To make the data sets available to your remote project named *POTCOB*, you will need to add them. For this lab we just want to add three datasets, but you could select to add specific members or the whole dataset.

5.2.1 ▶▶ Using the **Remote Systems** view (on your right). Expand the **MVS Files** and **My Data Sets** under **dallas** until you see the data sets that you allocated before.

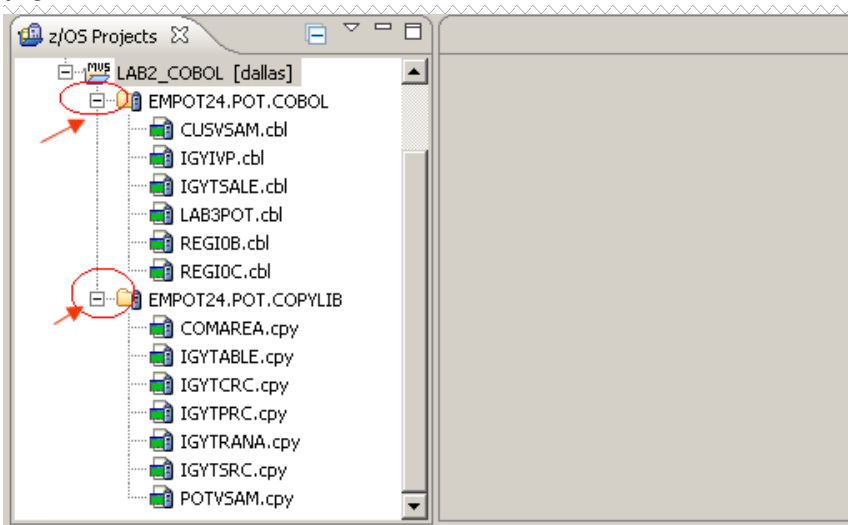
5.2.2 ▶▶ Select **EMPOTXX.POT.COBO**, **EMPOTXX.POT.COPLYLIB**, and **EMPOTXX.POT.JCL** (hold the **Ctrl** key for multiple select), Right-click mouse and select **Add To Subproject...**



5.2.3 ▶▶ On the **Add Resources** panel, select the z/OS project **POTCOB** from the drop down list (there is only one) and subproject **LAB2_COBO** from its drop down list (there is also only one) and click **Finish**.



5.2.4 ▶▶ Switch to the z/OS Projects view and expand **LAB2_COBO**. You will now see that **EMPOTXX.POT.COBO**, **EMPOTXX.POT.COPLYLIB** and **EMPOTXX.POT.JCL** are defined to the **POTCOB** project. Also expand **EMPOTXX.POT.COBO** and **EMPOTXX.POT.COPLYLIB**. The z/OS Projects view should look like this:



Show Dependencies

In this example we have added the COPYBOOKS in the project. This is not necessary. Rational Developer for System z has a nice feature named *Show dependencies*. You can automatically add the dependencies of a COBOL or PL/I program to your MVS subproject. The COBOL or PL/I program must be part of an MVS subproject, and this program must not contain any syntax errors.

Note: If your program depends on a file that cannot be found, an error message is returned and the file is skipped. To ensure that files are not skipped, you may first run a local syntax check on your program; if the syntax check does not produce any errors, then all the dependencies can be located and no files will be skipped. The sequence would be like the shown below. **DO NOT perform this on this lab to save time. This will send a job to z/OS to be executed..**



1. Right-click the COBOL or PL/I program in the z/OS® Projects view.
2. Select **Show Dependencies** in the context menu. A job will be submitted to z/OS and if the system is able to execute your job a window opens, listing the dependencies for the selected program. This operation could take minutes depending on the z/OS availability.
3. Select the dependencies you wish to add to your subproject in the list box.
4. Check the **Add selected to subproject** check box.
5. Click **Finish**.

The selected dependencies would be added to your MVS subproject.

What have you done so far?

At this point you have connected to z/OS and have submitted a job that allocated and copied some members required to do this lab. You have associated to MVS files configuration properties.

In section 4 you copied the CUSVSAM COBOL program from your local workstation to a z/OS dataset.

In section 5 you created a z/OS project and added three datasets to this project and you are ready to proceed.

If you disconnect now and connect later, you can go the z/OS projects and the code that you are working on is all grouped there making your work easier since you don't need to start looking for all your components.

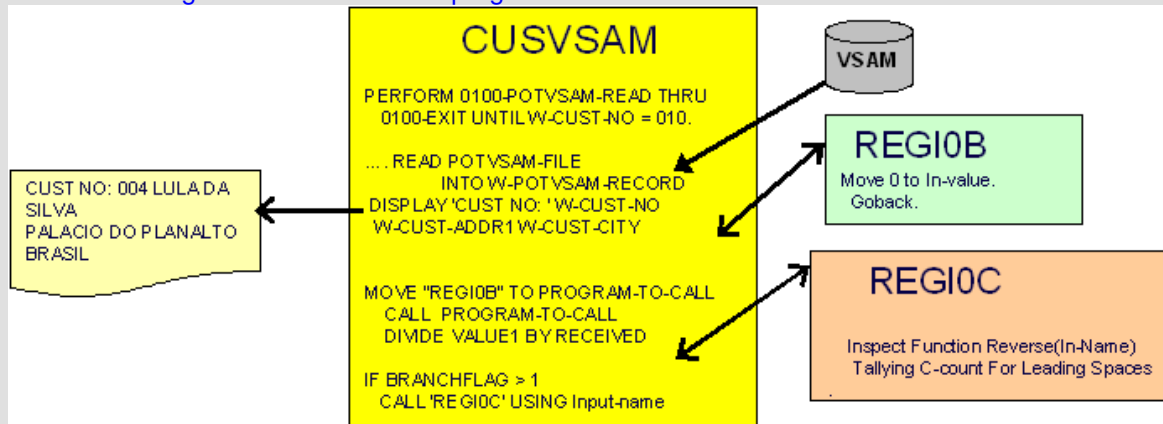


Section 6 – Working with z/OS remote assets

We will work with a z/OS member using the editor.

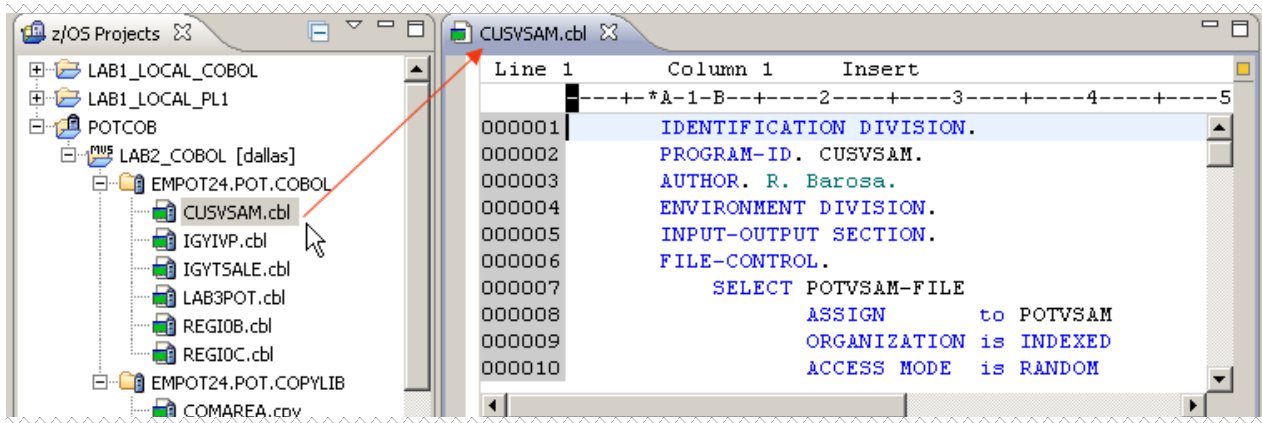
What z/OS remote assets you will work with?

You have copied a COBOL program named CUSVSAM from your workstation to a PDS member. This is a batch program that reads a VSAM data set and display it contents. Also this program does a Dynamic call and a Static call to two other COBOL programs named REGI0B and REGI0C. The figure below shows this program architecture:

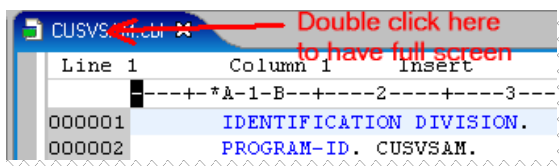


6.1 Editing a remote COBOL program

6.1.1 **▶▶** Using the **z/OS Projects** perspective and the **z/OS Projects** view, expand **EMPOTXX.POT.COBO** in the **POTCOB** project. Double click on **CUSVSAM.cbl** to open the file using the editor. You should see something similar to the following:

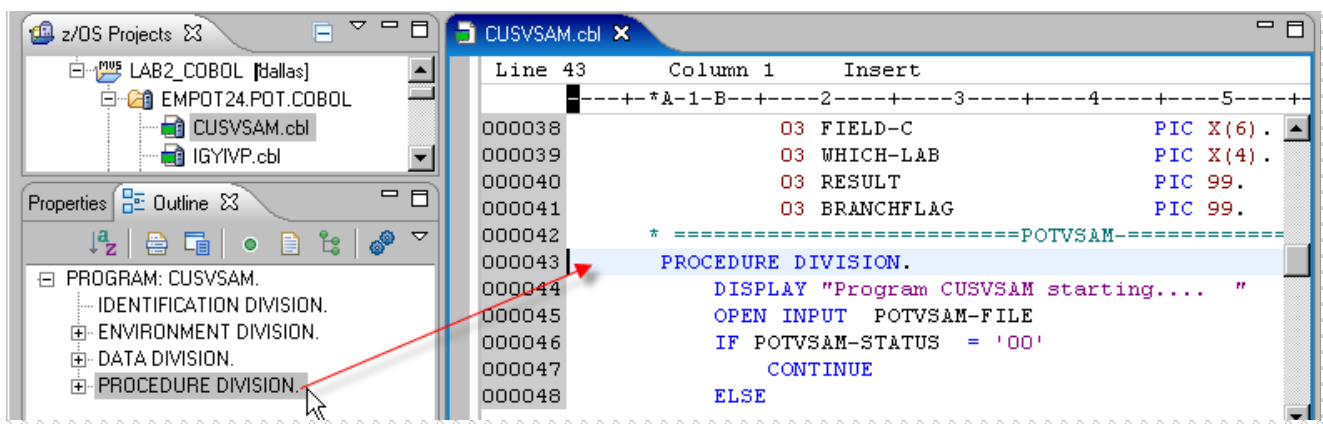


6.1.2 **▶▶** You can use the mouse to **expand the editor area** to see more lines of the COBOL program. Also remember that when double clicking in the title (**CUSVSAM.CBL**) you can either expand up to a full screen or return to original size. Also **Window → Reset Perspective** restores the default (in that case you may need to rearrange the Remote Systems and Team views on top of z/OS projects view again).



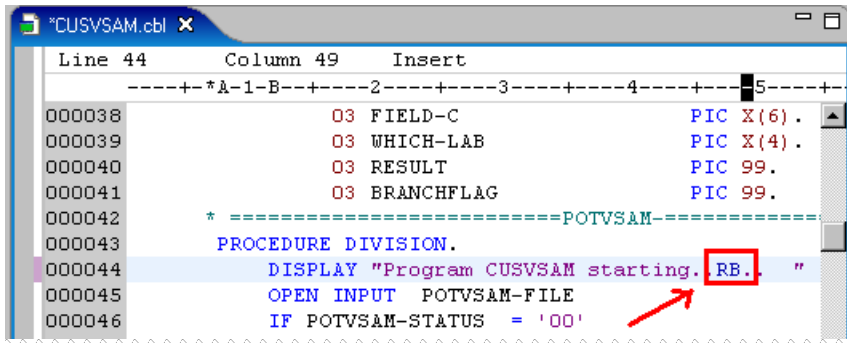
6.1.3 **▶▶** Click in the **Outline** tab to see the Outline view. Using the editor, browse the program and note that the contents of the outline view is synchronized with the COBOL source code and vice versa.

▶▶ Click on the **PROCEDURE DIVISION**

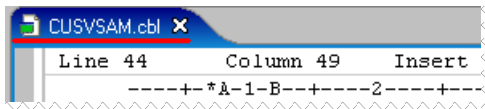


6.1.4. We will now do a small change in the DISPLAY statement.

▶▶ Locate the line 44 and add **XX** to the DISPLAY statement where XX could be your initials (RB in the picture below).



6.1.5 ▶▶ Save the change using the key combination **Ctrl + S**. The * next to the title will go away. Do not close the editor.



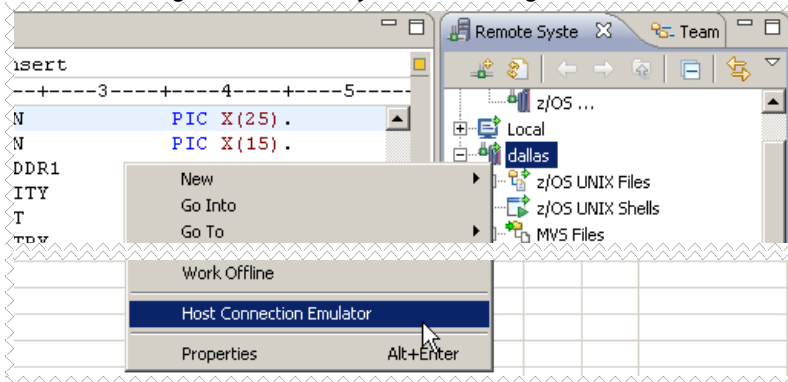
When you are editing a PDS member, this resource is locked to prevent multiple updates, if you are interested go to the TSO and verify this locking, execute the steps 6.2.1 thru 6.2.11 otherwise jump to the step 6.3.1

6.2 (Optional) Emulating TSO under Rational Developer for System z

If you are running late skip this step and go to step 6.3.

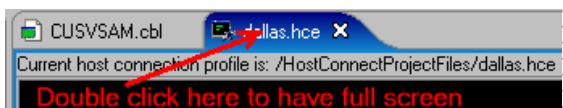
Now you will use TSO to logon to the same z/OS system that you are connected and verify that your changes were saved to the z/OS.

6.2.1 ▶▶ Using the Remote Systems view right-click on **dallas** and select **Host Connection Emulator Support**.

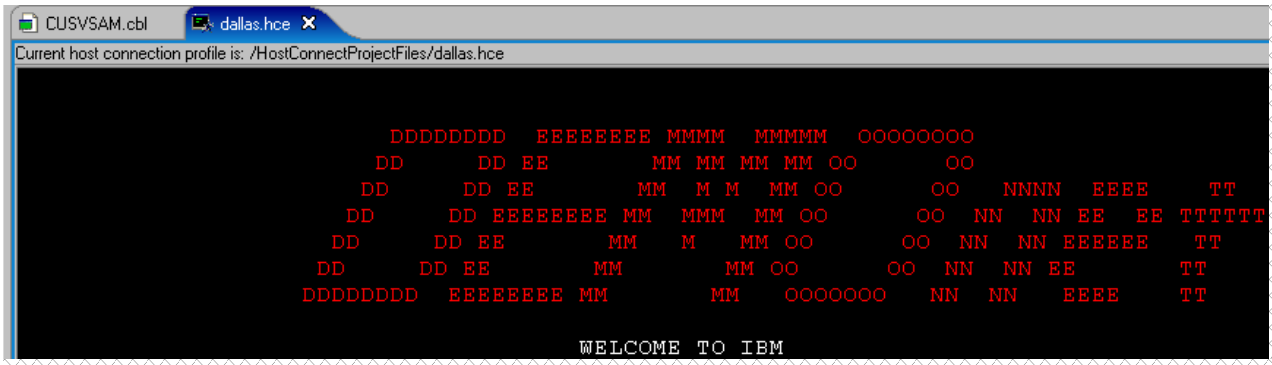


You will be emulating a 3270 screen on the Dallas z/OS system,

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



You will have more space to see the 3270 emulation

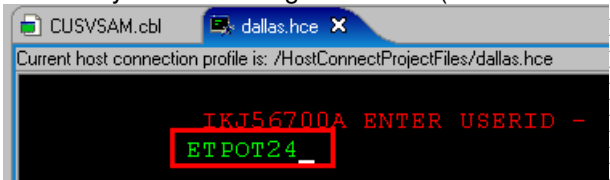


▶▶ Type **TSO** in the DEMONET black screen and press **Enter** (use the **Right CTRL** key as the enter key)

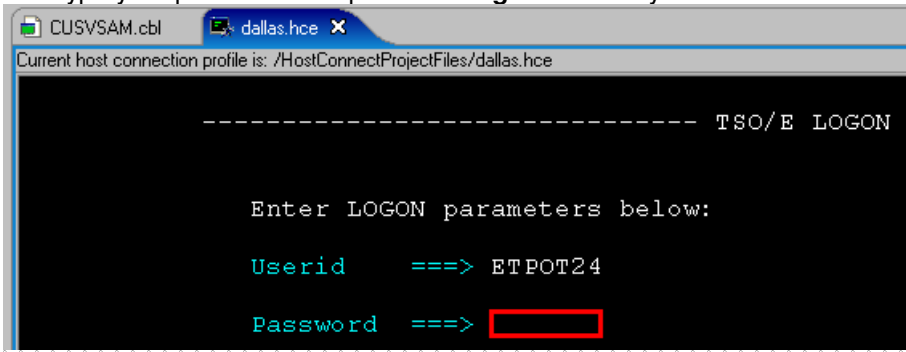


To log on to the TSO :

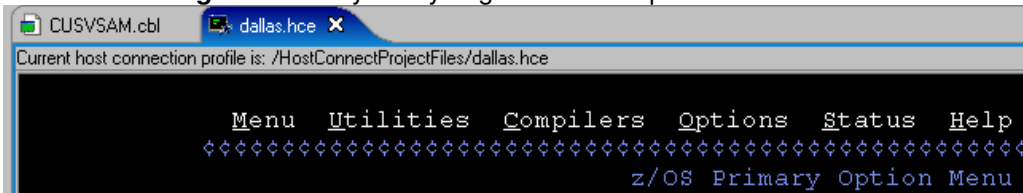
▶▶ Use your z/OS assigned user id (like EMPOTXX below) and press the **Right Ctrl** key.



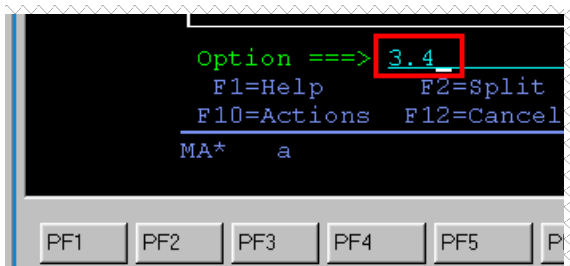
▶▶ Type your password and press the **Right CTRL** key



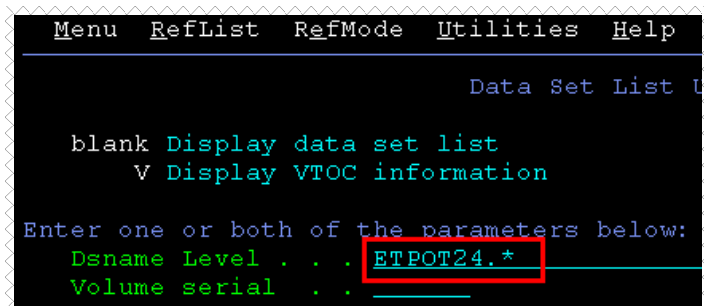
▶▶ Press the **Right CTRL** key until you get the z/OS option menu:



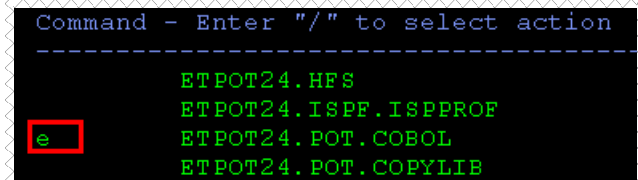
6.2.2 We want the ISPF Utilities... Type **3.4** in the command line and press the **Right CTRL** key:



6.2.3 Using the Utility panel type your **userid.*** (like EMPOTXX.*) to see the datasets that start with that high qualified name and press the **Right CTRL** key:



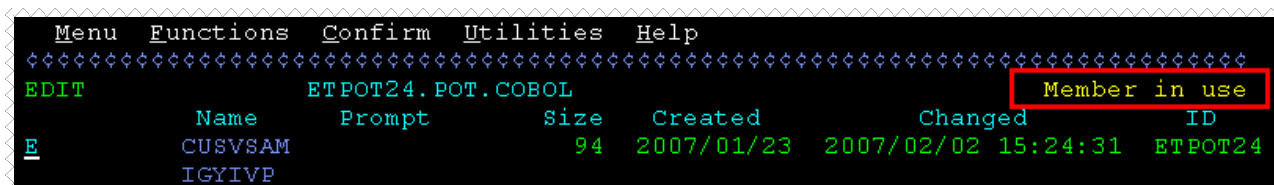
6.2.4 Type **E** in the dataset **EMPOTXX.POT.COBO** and press the **Right CTRL** key



6.2.5 Type **S** to select the member **CUSVSAM** and press the **Right CTRL** key to edit it.



Since you are editing this file with Rational Developer for System z (assuming you did not close the editor) you will have a message that tells you that the member is in use.. Cool ah? Otherwise people could edit the file being used by Rational Developer for System z.



6.2.6 **▶▶** Press **F1** twice and you will see that this file is being edited by **FEKFLK00**

```
Data set 'ETPOT24.POT.COBO (CUSVSAM) '
is in use by the following 1 user(s) and/or job(s):
-----
FEKFLK00
```

6.2.7 **▶▶** Press **F3** to exit and type **B** in place of E to browse the file..

```
EDIT          ETPOT24.POT.COBO
Name          Prompt          Size
b            CUSVSAM          94
TCXVLR
```

6.2.8 **▶▶** Use **F8** and **F7** to locate the modified statement as seen below:

```
PROCEDURE DIVISION.
    DISPLAY "Program CUSVSAM starting..RB.. "
    OPEN INPUT POTVSAM-FILE
```

6.2.9 Logoff the TSO.

▶▶ Press **F3** five times

▶▶ Type **2** to delete data sets if requested

What have you done so far?



At this point you have connected to z/OS. In this section 6.2 that was optional you emulated a z/OS 3270 session, logged on to TSO, and tried to edit the same member that you are using in Rational Developer for System z - but without success. However, you could browse this member since browse is a read-only operation and then check the change that you had made in the member.. You are ready to continue..

6.2.10 **▶▶** Type **LOGOFF** and press the **Right Ctrl** key.

The LOGOFF is important since if you stay logged a timeout will disconnect you and you will NOT be able to logon again.

```
CUSVSAM.cbl  dallas.hce x
Current host connection profile is: /HostConnectProjectFiles/dallas.hce
ETPOT24.SPFLOG1.LIST has been deleted.
READY
logoff_
```

6.2.11 **▶▶** Close the TSO emulator window clicking on **x**.

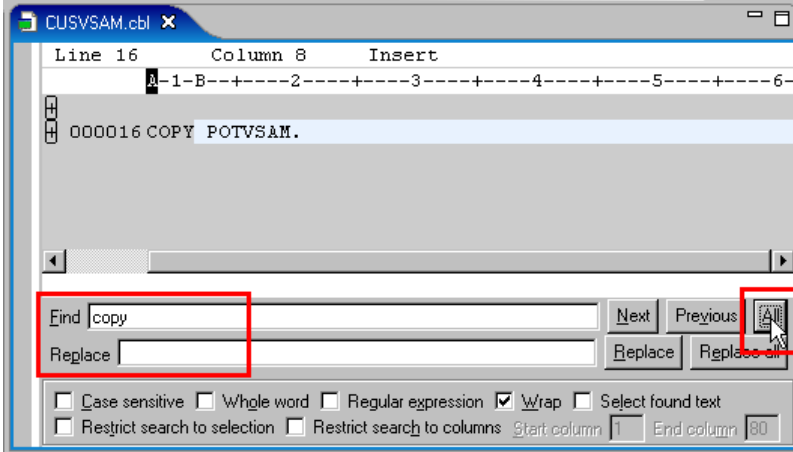
```
REGIOA.cbl  demomvs.hce x
```

6.3 Exploring the use of copybook expansion

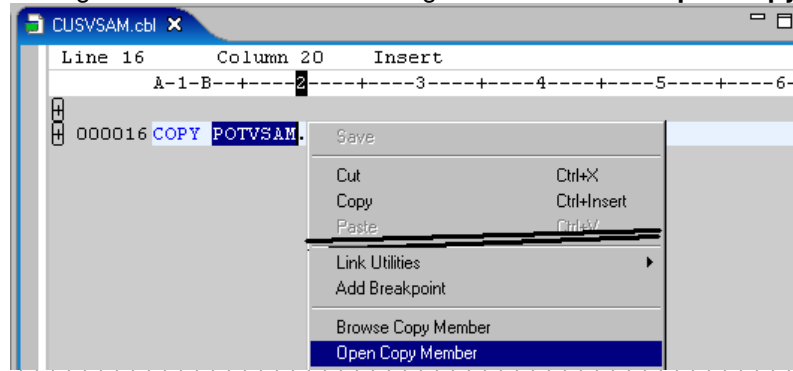
When we are editing a COBOL program sometimes we need to see the contents of a copybook. This function is available on the editor and might help you. See one example below.

6.3.1 ▶▶ Still editing the program **CUSVSAM.cbl** press **CTRL + F** to bring up the Find/Replace dialog (shown below). Enter **copy** in the Find field and click the **All** button.

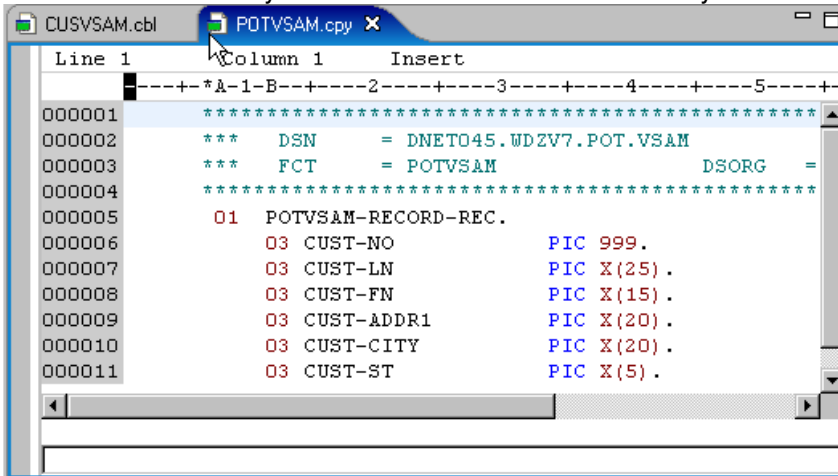
You should see something similar to the following. All lines with 'copy' will be shown. (we have only one)



6.3.2 ▶▶ Double-click on the copybook name **POTVSAM** (line 16) to highlight it. This can be done by double-clicking on the word POTVSAM. Right-click and select **Open Copy Member**



This will open the **POTVSAM.cpy** copybook in another editing window as shown below. Remember that this asset is also on the z/OS. If you move the mouse to the blue title you will see where this copybook is located:



How the copy book is found?

Copybooks are resolved based on the value of Copy Libraries in the properties defined in the COBOL Settings as seen below.

The copybooks are found even though they are NOT part of your MVS Subproject. Usually there is no sense to add copybook libraries to MVS Subprojects, since they are used for many projects (unless they also need updates).

Note that in this lab we added the copybook but this is not necessary,

One option *Show Dependencies* will help to identify possible resources necessary for a COBOL program,.



Cobol Compile Step Options

Compile Procedure Name: ZLAXFCOC

Compile Procedure Step Name: COBOL

Compiler Options:

Listing Output Data Set: <HLQ>.POT.LISTING

Debug Data Set:

Object Deck Data Set: <HLQ>.POT.OBJ

Copy Libraries: <HLQ>.POT.COPYLIB

Support Error Feedback

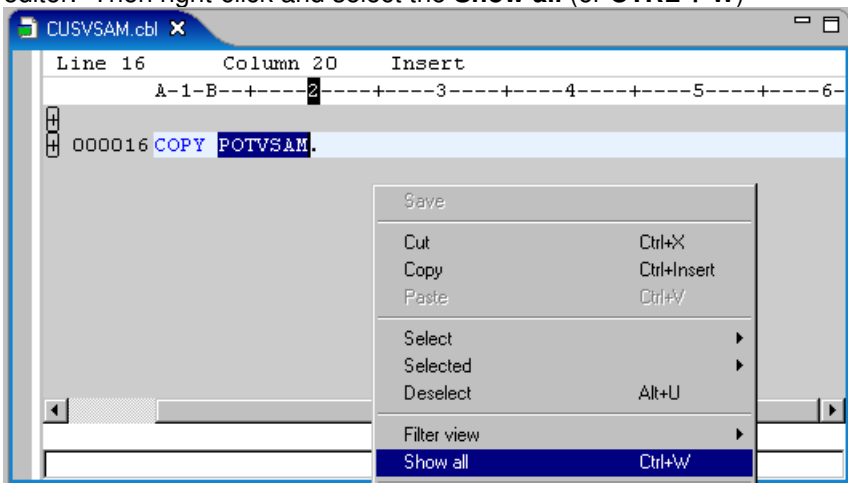
Data Set Qualifier for Compiler Errors: <HLQ>.POT.LISTING

Additional JCL:

6.3.3 Close the **POTVSAM.cpy** editor panel clicking on the X..



6.3.4 To expand the program again, first click anywhere on the **CUSVSAM.cbl** editor area to make it the active editor. Then right-click and select the **Show all** (or **CTRL + W**)



This action removes the filtered view (currently showing all statements with the word **copy**) and displays all statements.

6.4 Modify the COBOL Program

Program CUSVSAM.

This is a batch program that reads all records from a VSAM KSDS file and prints them using DISPLAY.

Also this program calls two other COBOL programs (REGI0B and REGI0C) using dynamic and static calls.

The subroutine that is called dynamically does a division by zero that will cause an abend with System Completion Code=0CB.

Using the z/OS Debug we can intercept the abend, modify the value to be other than zero, go back to the division statement and re-execute the division avoiding the abend.



6.4.1 Using the **Outline** view, navigate to the first **Procedure Division** statement by clicking on it.

```
Line 43      Column 1      Insert
-----+---+-----+---+-----+---+-----+---+-----+---+-----+
000038              03 FIELD-C              PIC X(6) .
000039              03 WHICH-LAB            PIC X(4) .
000040              03 RESULT                PIC 99 .
000041              03 BRANCHFLAG           PIC 99 .
000042      * =====POTVSAM=====
000043      PROCEDURE DIVISION.
000044      DISPLAY "Program CUSVSAM starting.... "
000045      OPEN INPUT POTVSAM-FILE
000046      IF POTVSAM-STATUS = '00'
000047          CONTINUE
000048      ELSE
```



Strange behavior using Outline?


If clicking in the outline do not cause the positioning as it should, close the editor and open it again. This is a known issue that will be fixed soon.

6.4.2 We want to execute this code, but 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.

Using the outline view, find the PROCEDURE DIVISION and Go to line 44 of **CUSVSAM.cbl** (or, you could use the command **CTRL + L** and type 44).

6.4.3 Lets introduce a small error.

Change **DISPLAY** with **DSPLAY** to force an error. An yellow mark  shows that something is wrong. Move the mouse to the yellow mark to see what the error is. This is a new feature in version 7.5.

```
Line 44      Column 13      Insert
-----+---+-----+---+-----+---+-----+---+-----+---+-----+
000038              03 FIELD-C              PIC X(6) .
000039              03 WHICH-LAB            PIC X(4) .
000040              03 RESULT                PIC 99 .
000041              03 BRANCHFLAG           PIC 99 .
000042      * =====POTVSAM=====
000043      PROCEDURE DIVISION.
000044      DSPLAY "Program CUSVSAM starting..RB.. "
000045      OPEN INPUT POTVSAM-FILE
000046      IF POTVSAM-STATUS = '00'
000047          CONTINUE
000048      ELSE
```

6.4.4 Go to the **line 75** to change the IF statement (under paragraph 0200-LOGIC)

```
*CUSVSAM.cbl
Line 75      Column 1      Insert      4 changes
-----+*A-1-B-----2-----3-----4-----5-----
000071      MOVE 'BBBBBB' to FIELD-B.
000072      MOVE 'CCCCCC' to FIELD-C.
000073      MOVE "LAB2" to WHICH-LAB.
000074      0200-LOGIC.
000075      IF WHICH-LAB = 'LAB2'
000076      * If is LAB2 lets do a dynamic CALL.. and force
000077      MOVE "REGIOB" TO PROGRAM-TO-CALL
000078      CALL PROGRAM-TO-CALL USING RECEIVED-F
000079      MOVE 66 TO VALUE1
000080      DIVIDE VALUE1 BY RECEIVED-FROM-CALLED
000081      DISPLAY "The result is ... " RESULT
```

Change from 'LAB2' to 'NODYNAM'. Do NOT save the changes.

```
000073      MOVE "LAB2" to WHICH-LAB.
000074      0200-LOGIC.
000075      IF WHICH-LAB = 'NODYNAM'
000076      * If is LAB2 lets do a dynamic CALL.. and force
000077      MOVE "REGIOB" TO PROGRAM-TO-CALL
```

6.5 Using Local Syntax Checking

6.5.1 We want to compile the COBOL program using the local compiler. This will save some CPU in the z/OS system.

Note that even though the assets are remote we can use the local compiler and save some CPU on the z/OS. But first we need to save the changes. There is one option that performs both actions.

Right-click and select **Save and Syntax Check → Local**

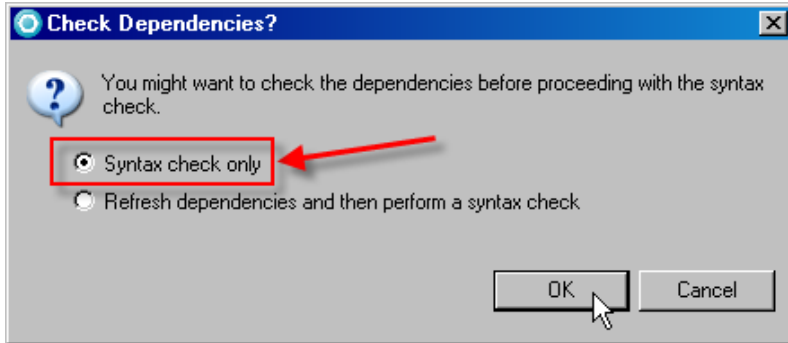
```
*CUSVSAM.cbl
Line 70      Column 40      Insert      1 change
-----+*A-1-B-----2-----3-----4-----5-----
000070      MOVE 'AAAAAA' to FIELD-A.
000071      MOVE 'BBBBBB' to FIELD-B.
000072      MOVE 'CCCCCC' to FIELD-C.
000073      MOVE "LAB2" to WHICH-LAB.
000074      0200-LOGIC.
000075      IF WHICH-LAB = 'NODYNAM'
000076      * If is LAB2 lets do a dynamic CA
000077      MOVE "REGIOB" TO PROGRAM-
000078      CALL PROGRAM-TO-CALL USI
000079      MOVE 66 TO VALUE1
```

Context Menu:

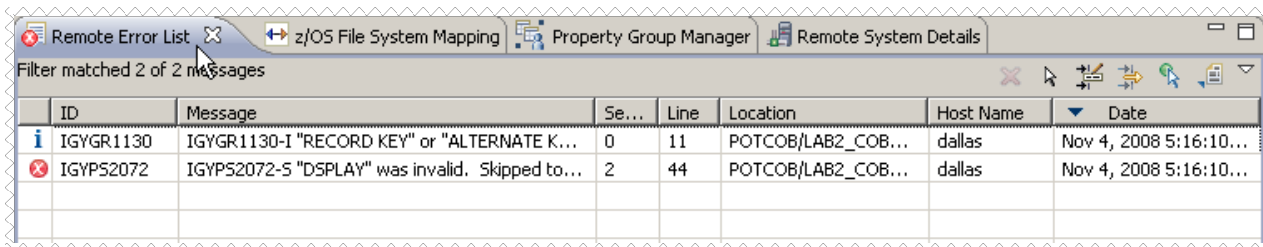
- Save
- Cut (Ctrl+X)
- Copy (Ctrl+Insert)
- Paste (Ctrl+V)
- Select
- Selected
- Save and Syntax Check
 - Local
 - Remote
- Content assist (Ctrl+Space)

6.5.2 **▶▶** Select **Syntax check only** since there were no changes in any of the dependencies (like the copybooks.) and click **OK**

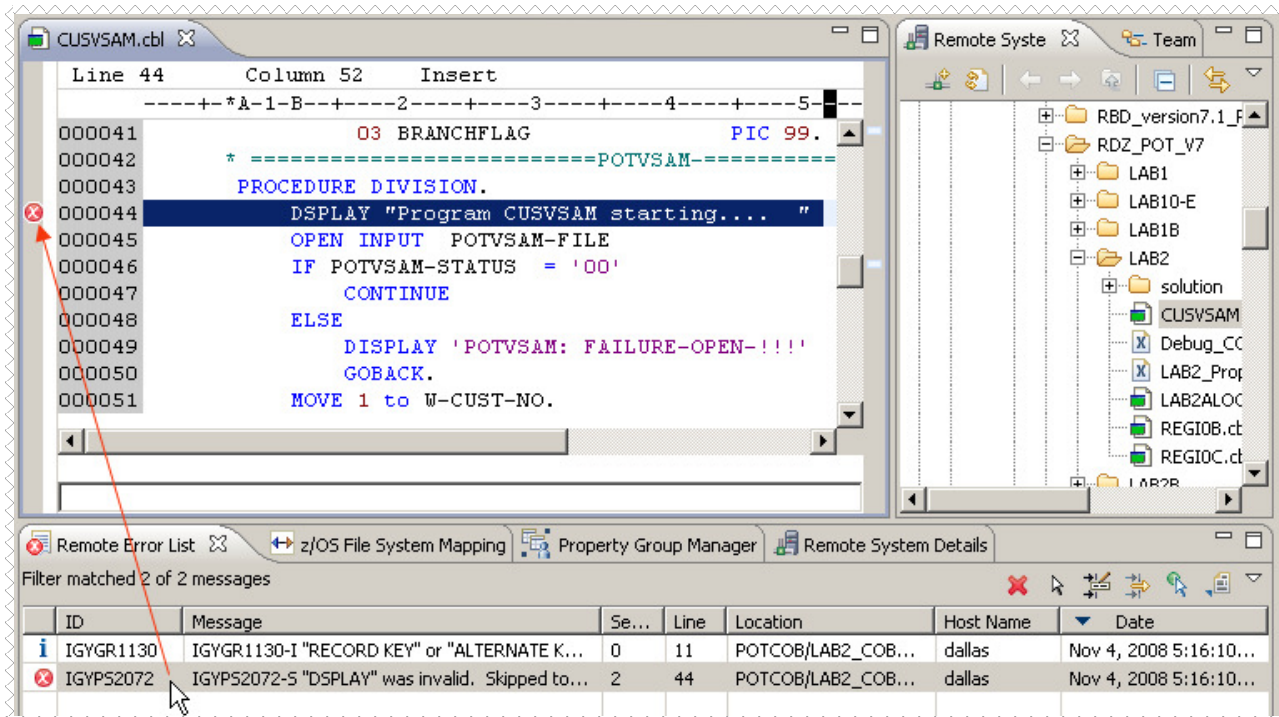
IMPORTANT → Do not use the default selection (**Refresh dependencies...**) this might take a while. If you did by accident be patient. This will submit a JOB to z/OS...



6.5.3 **▶▶** Click on **Remote Error List** view (in the bottom) to check syntax errors.



6.5.4 **▶▶** **Double click** on the error message. This should bring you to the editor positioned at line 44.



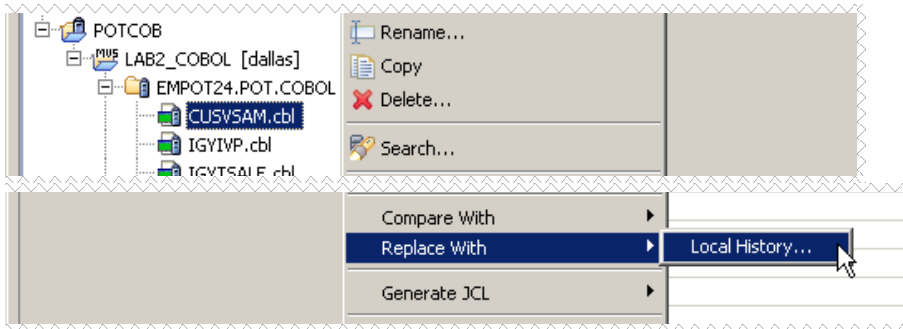
6.6 Using Replace with Local History

6.6.1 You will fix that by returning to the old version that had the correct DISPLAY statement.

▶▶ Click **CTRL + Shift + F4** to close any opened editor.

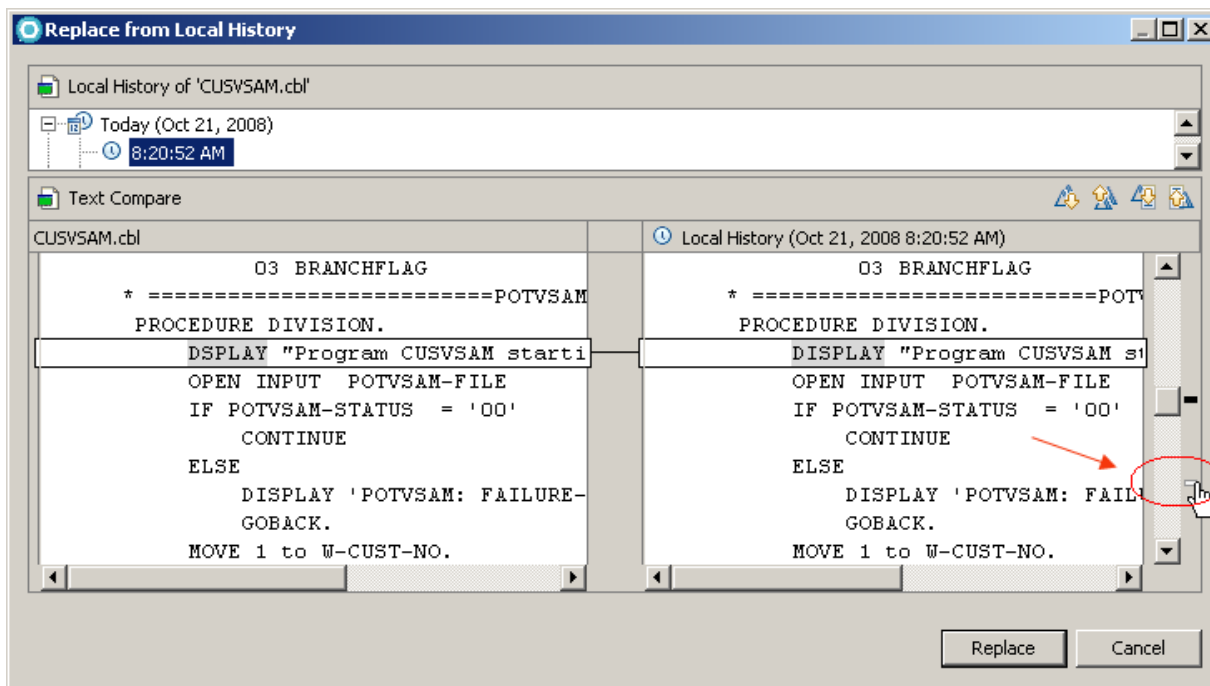
6.6.2 A nice feature of Rational Developer for System z is the capability to recover previous versions (even remote code) using the local workstation. This is very useful when you delete and save components on the z/OS where undo is not possible after you have saved the changes.

▶▶ Right click on **CUSVSAM.cbl** and select **Replace With → Local History**



6.6.3 This operation can take a while since it goes to z/OS. It will show all the previous versions (you have just one).

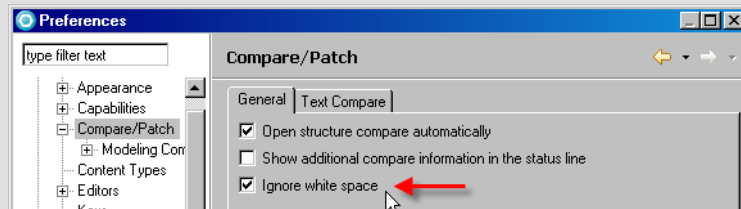
▶▶ Click on the **white spot** on the right (see the little hand in the figure below) to see the next change.



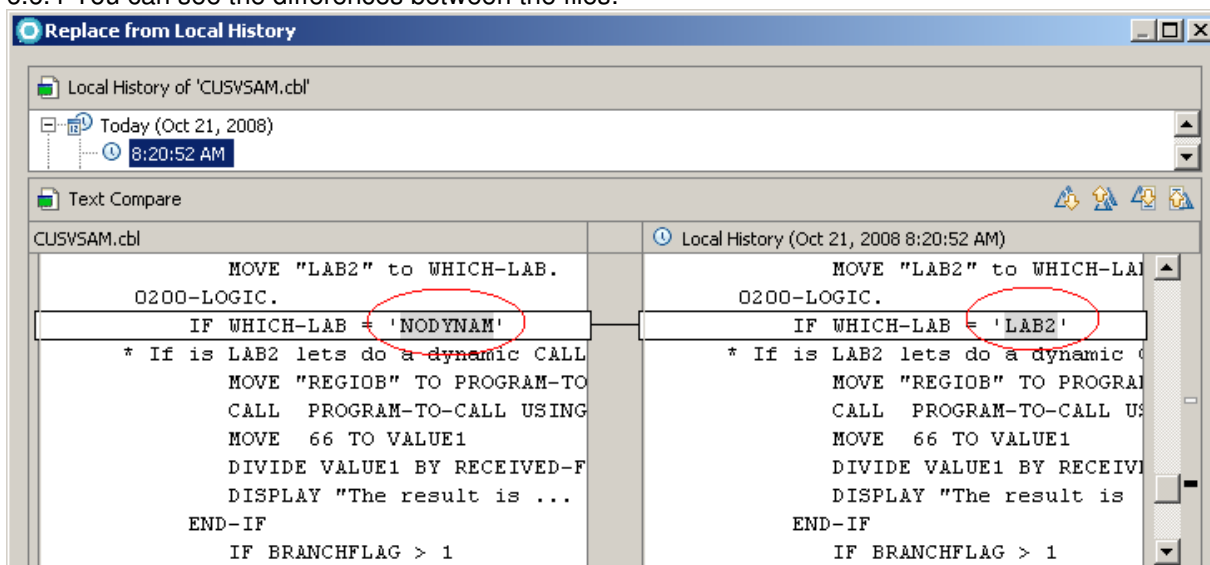
If you have a different result than shown above you might have change the preferences on your workspace. Also be aware that you are not using the suggested workspace for the labs and might have future issues. You can continue for now, but in the next lab switch to the correct workspace.

To change the preferences:

1. Select **Windows** → **Preferences**
2. **Expand General** and click on **Compare/Patch**
3. Select **Ignore white space**
4. Click **OK**.

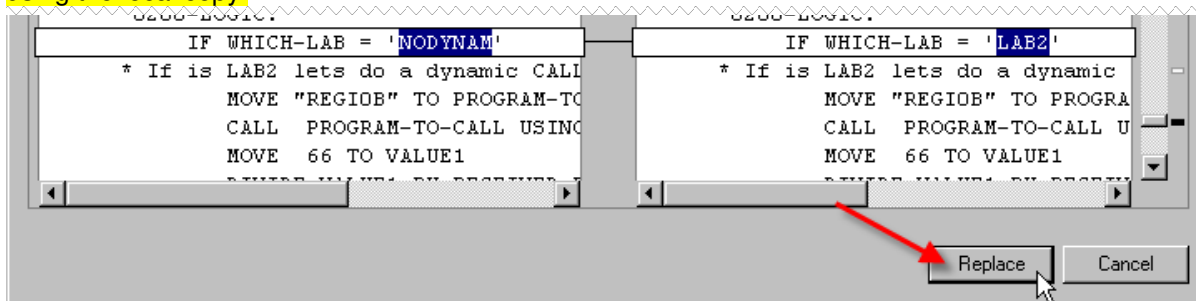


6.6.4 You can see the differences between the files:



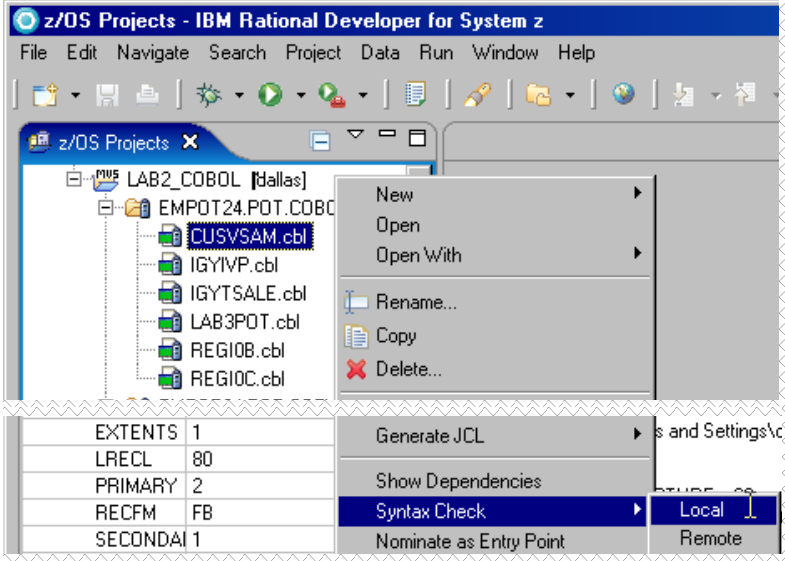
6.6.5 **▶▶** To return to the previous version (the original without changes) click on **Replace** button. The version without changes will return to the z/OS system.

Note all operations are being done at the z/OS. Depending on the network this could be slower than if you were using the local copy.

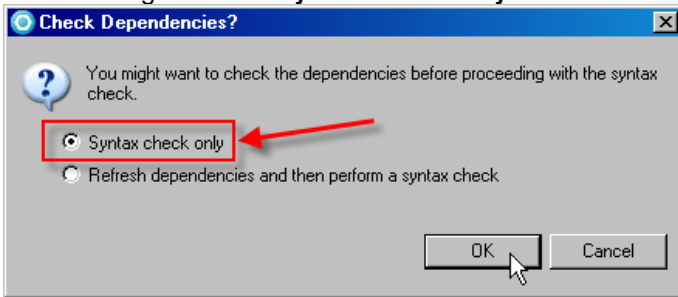


6.6.6 Perform another Local Syntax Check

▶▶ Right click on **CUSVSAM.cbl** and select **Syntax Check → Local**



6.6.7 ▶▶ Again select **Syntax check only** and click **OK**



6.6.8 Click in the Remote Error List view and note that we have a warning now in the Remote Error List. The error message from the last time is gone.

6.6.9 **Double click** in the warning and you will be positioned at the location of the warning. We can ignore warnings for the purpose of this demo.

The screenshot shows a COBOL editor window with a warning message highlighted. The warning message is:

```

000011 RECORD KEY is CUST-NO
000012 FILE STATUS is POTVSAM-STATUS.
  
```

The editor also shows the following code:

```

000006 FILE-CONTROL.
000007     SELECT POTVSAM-FILE
000008     ASSIGN      to POTVSAM
000009     ORGANIZATION is INDEXED
000010     ACCESS MODE is RANDOM
000011     RECORD KEY is CUST-NO
000012     FILE STATUS is POTVSAM-STATUS.
000013 DATA DIVISION.
000014 FILE SECTION.
000015 FD POTVSAM-FILE.
000016 COPY POTVSAM.
  
```

The warning message is highlighted in the editor, and a red arrow points to it. The warning message is also visible in the Remote Error List at the bottom of the window.

The Remote Error List shows the following message:

ID	Message	Se...	Line	Location	Host Name	Date
IGYGR1130	IGYGR1130-I "RECORD KEY" or "ALTERNATE K...	0	11	POTCOB/LAB2_COB...	dallas	Nov 4, 2008 5:28:09...

The Remote System Explorer on the right shows the following structure:

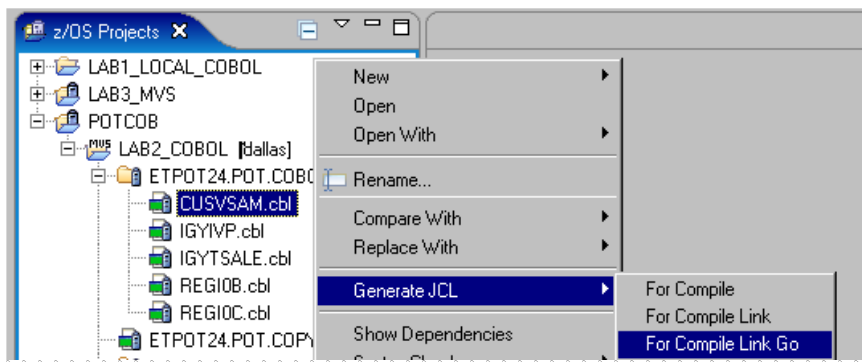
- RBD_version7.1_f
 - RDZ_POT_V7
 - LAB1
 - LAB10-E
 - LAB1B
 - LAB2
 - solution
 - CUSVSAM
 - Debug_CC
 - LAB2_Prog
 - LAB2ALOC
 - REGIOB.ct
 - REGIOC.ct
 - LAB2B

6.6.10 **Close** all opened editors if still opened. (**CTRL + Shift + F4**)

6.7 Generate JCL to compile, link and GO **without z/OS debugger**

6.7.1 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.

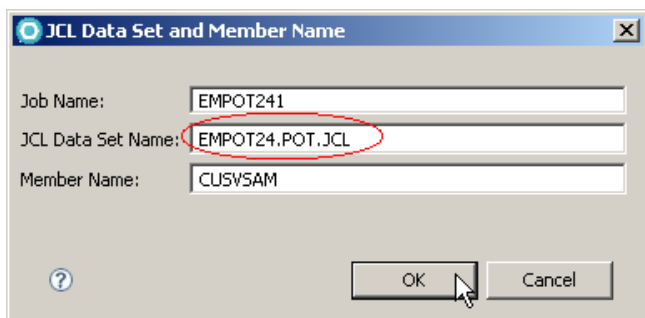
Using the *z/OS projects* View, right-click on **CUSVSAM.cbl** and select **Generate JCL → For Compile Link Go**.



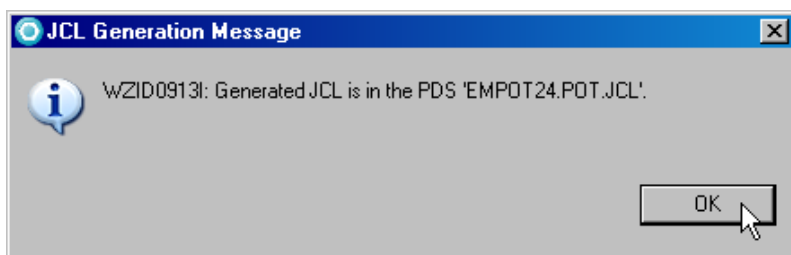
6.7.2 On the *JCL Data Set and Member Name* window, notice that the *JCL Data Set Name* is set to the value you specified for your project settings.

Be sure that the data set name is EMPOTXX.POT.JCL (where EMPOTXX is your userid) If not change to this data set name.

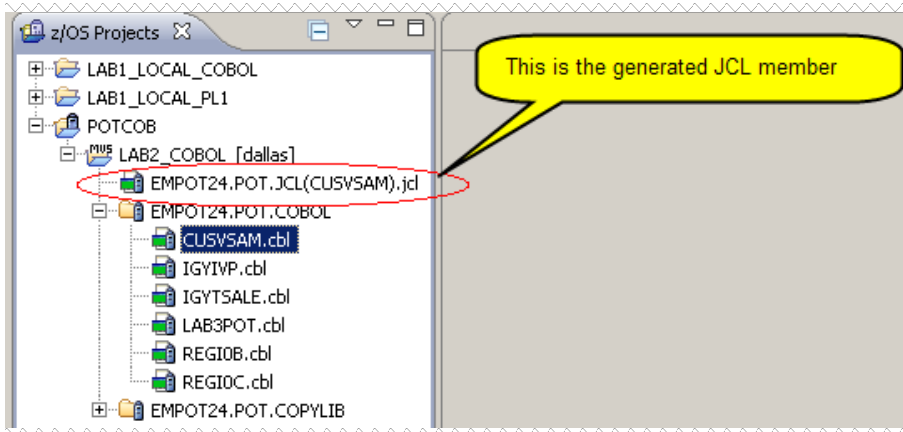
Click **OK**.



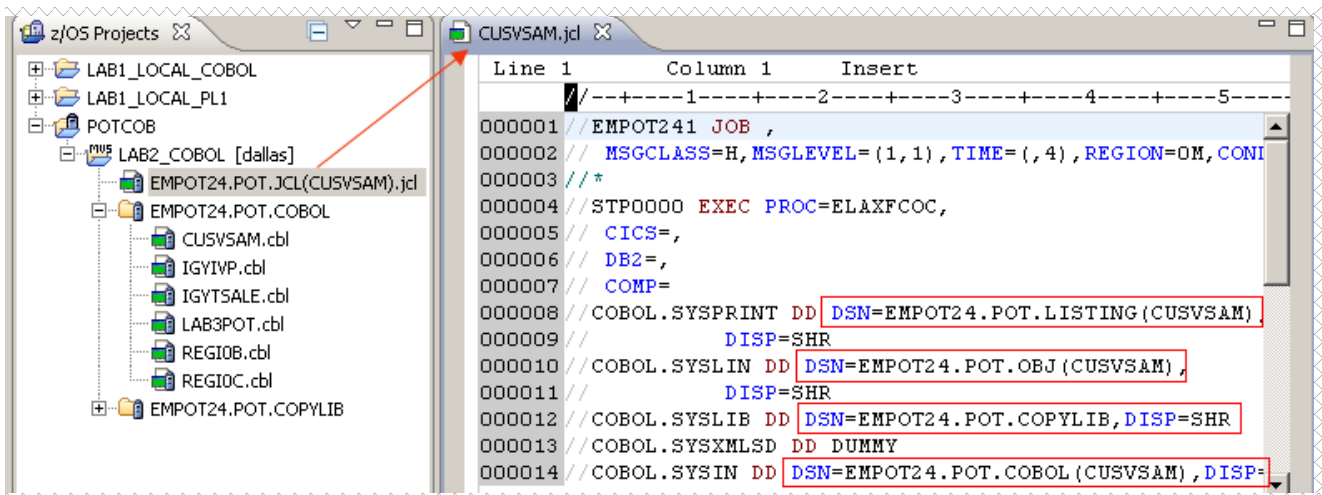
6.7.3 You should see the message below. Click **OK**.



6.7.4 Go to your z/OS Projects view, and you will see that **CUSVSAM.jcl** was generated.

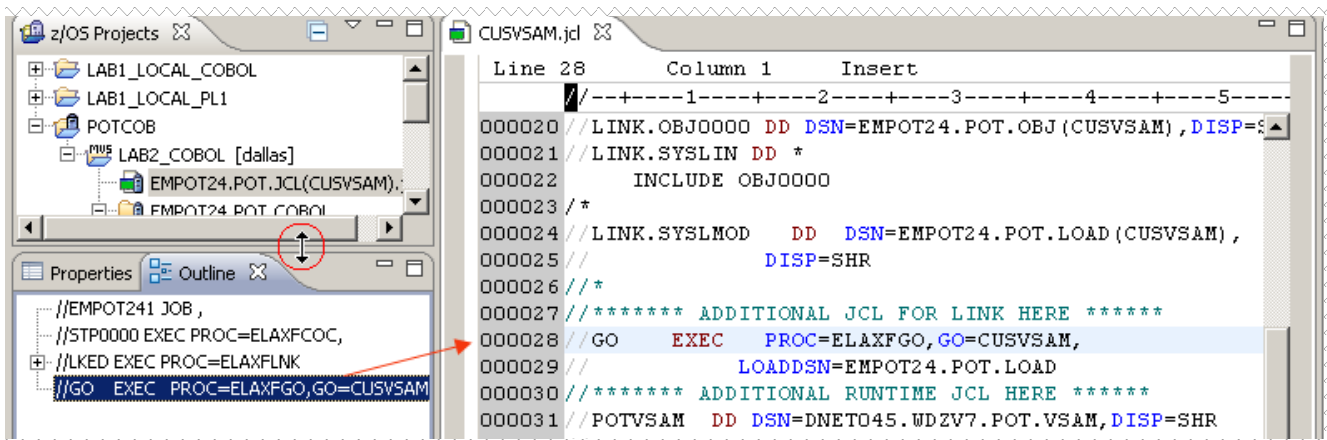


6.7.5 **▶▶** Open the JCL member (**double click** on it) and check the JCL generated. Also note that instead of EMPOT24.* you would have your user ID. The first 14 lines should look something like this.

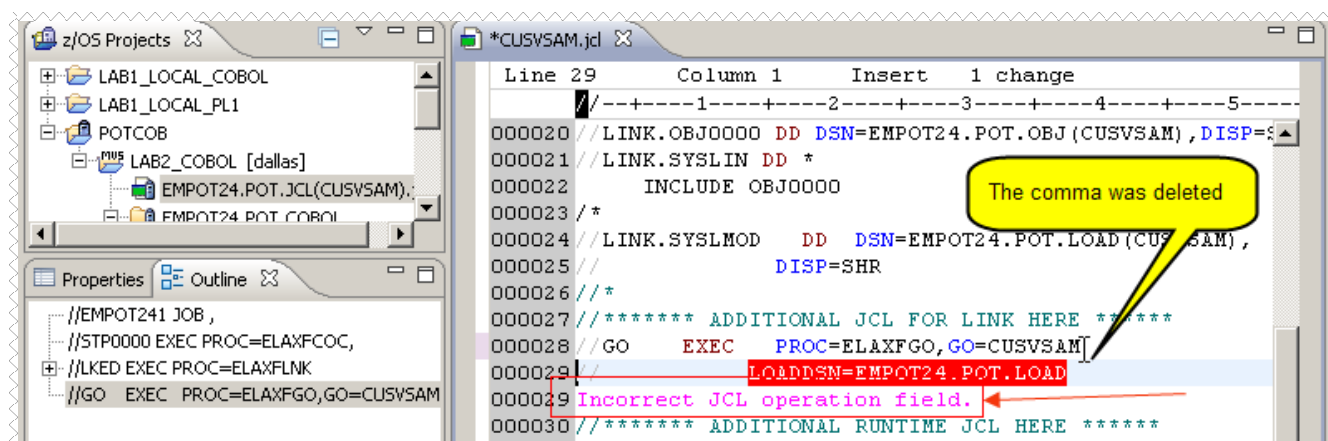


6.7.6 Notice that LPEX editor recognizes JCL and you can navigate your job through the *Outline* view.

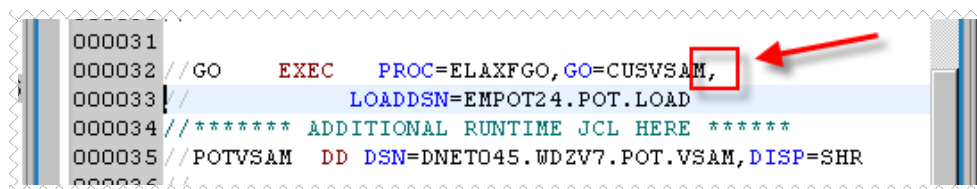
▶▶ Click on step **//GO** and note that the cursor will be at this location.



6.7.7 **▶▶** Add a JCL error, for example if you delete the **comma** in the end of the line as shown below (after **CUSVSAM**) and press **enter** you will have the error below. Note also that some keywords are recognized and displayed in **blue**.



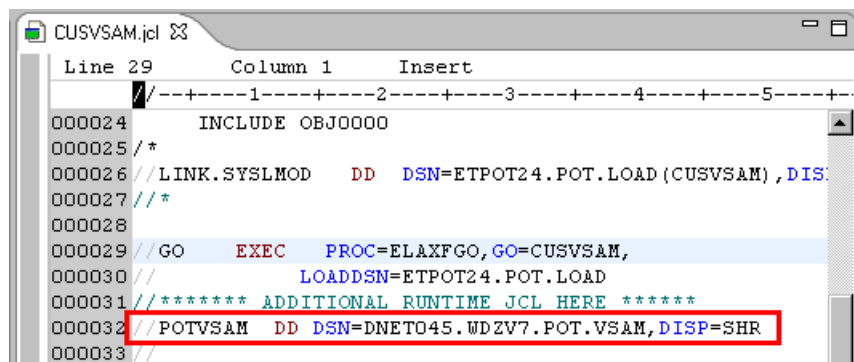
6.7.8 **▶▶** Fix the line **32** by adding a comma after **CUSVSAM** and pressing **enter**.



6.7.9 Take a look in the generated JCL.

▶▶ Using the *Outline view*, if not there already, **click** on the step **//GO**. Note the JCL card **POTVSAM** that is generated. This card was defined in the project properties as shown in the step 3.1.8.

The program *CUSVSAM* calls 2 other COBOL programs (*REGI0B* and *REGI0C*). To facilitate the labs, those programs were already compiled and are in the dataset *EMPOTXX.POT.OBJ* and *EMPOTXX.POT.LOAD* since they need to be included by the linkage editor.

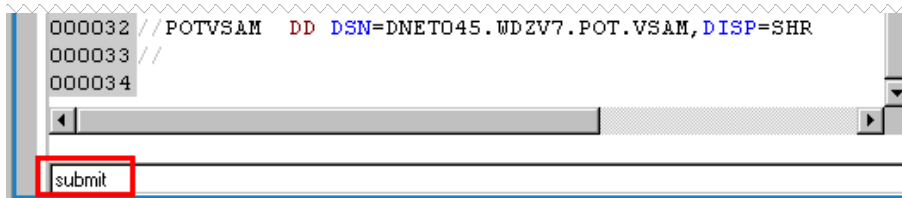


6.7.10 **▶▶** **Save** the changes (**Ctrl + S**). Do NOT close the editor.

6.8 Submit JCL for execution

6.8.1 Now you can submit the job to be run on the z/OS system. You can use the editor command or the context menu actions. Note that the JES subsystem must be connected, otherwise a message will indicate that submit was not accepted.

▶▶ Using the command editor type **submit** (or sub) and press **ENTER** as shown below; just as you could do it using TSO/ISPF



```
000032 //POTVSAM DD DSN=DNET045.WDZV7.POT.VSAM,DISP=SHR
000033 //
000034
submit
```

6.8.2 ▶▶ A message with your JOBID number will indicate that the job was accepted as seen below



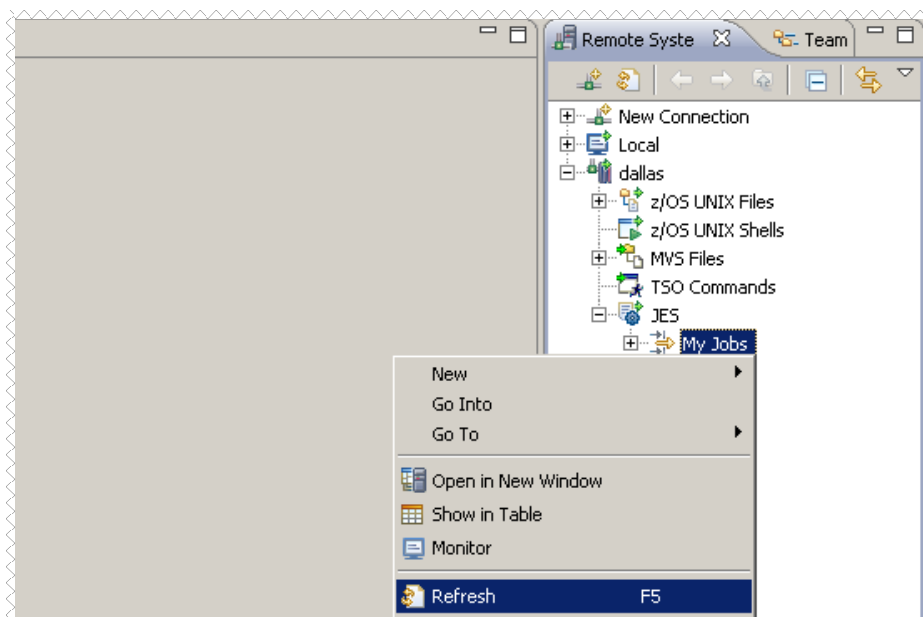
```
JOBID: JOB07853
```

6.8.3 ▶▶ Close the JCL editor by clicking on the  .

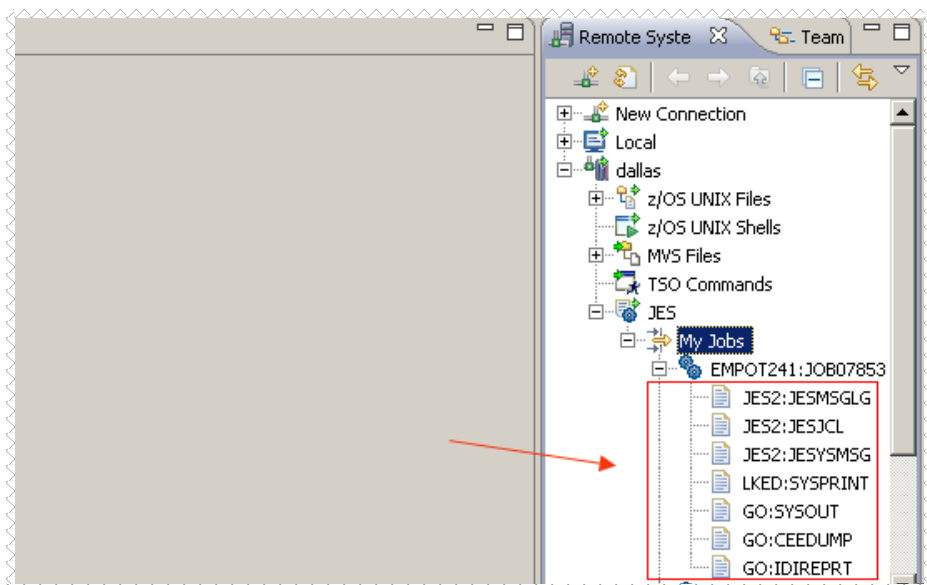
6.9 Access the output listings

You can check the job generated output listings,

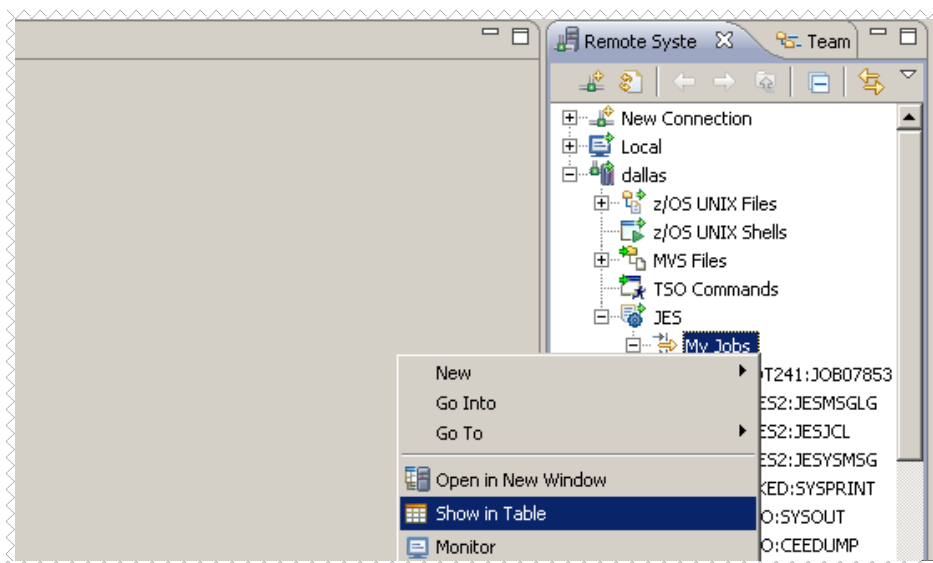
6.9.1 ▶▶ Using the *Remote Systems* view locate the node **dallas**, expand **JES** node
▶▶ Right-click on **My Jobs** and select **Refresh**.



6.9.2 ▶▶ Expand **My Jobs** and the first job on the queue and you will see the execution results



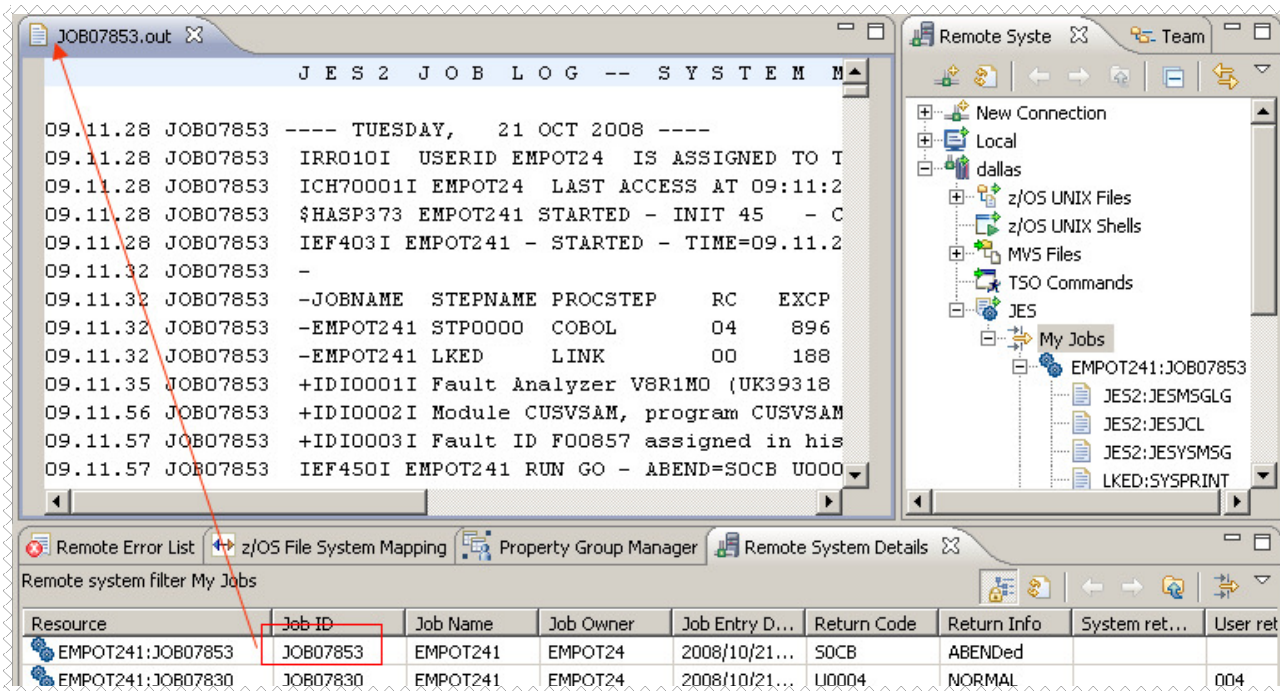
6.9.3 ▶▶ Right-click on **My Jobs** and select **Show in Table**. This is a good way to see the listing since you get more details, the return code, the dates, etc..



6.9.4 As we expected we had an ABEND and the return code is **0CB** (division by zero)

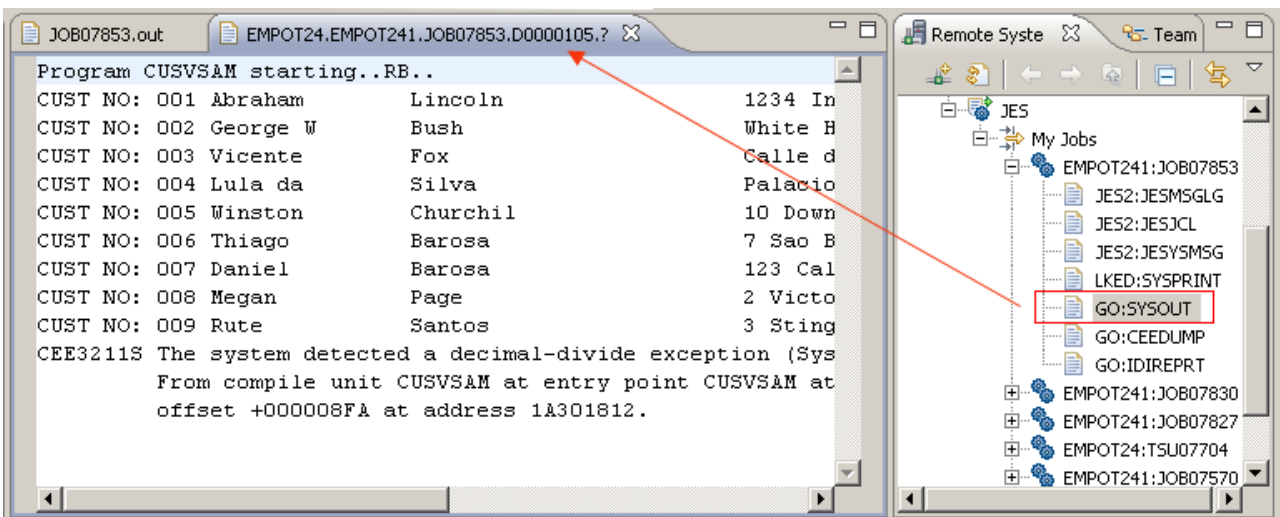
Resource	Job ID	Job Name	Job Owner	Job Entry D...	Return Code	Return Info	System ret...	User ret
EMPOT241:JOB07853	JOB07853	EMPOT241	EMPOT24	2008/10/21...	0CB	ABENDeD		nn4
EMPOT241:JOB07830	JOB07830	EMPOT241	EMPOT24	2008/10/21...	U0004	NORMAL		nn4

6.9.5 **▶▶** Using the Remote System Details, **double-click** on the job that you have submitted



6.9.6 Since the abend was after the display of the VSAM records we can see the records displayed.

▶▶ Double click on the step **GO SYSOUT** to the results of this step. Each record was read from the VSAM and displayed to the listing. You also can see the decimal-divide exception that caused the 0CB.



6.9.7 **▶▶** Close all the editors (**CTRL +Shift + F4**).

6.9.8 **▶▶** Using *Remote System Details* view select the jobs to be purged (use CTRL key if more than one), right-click and select **Purge**. The job(s) listing(s) will be purged if you have authorization allowing you to do so.

Resource	Job ID	Job Name	Job Owner	Job Entry D...	Return Code	Return Info	System ret...	User ret
EMPOT241:JOB07853	JOB07853			2008/10/21...	S0CB	ABENDed		
EMPOT241:JOB07830	JOB07830			2008/10/21...	U0004	NORMAL		004
EMPOT241:JOB07827	JOB07827			2008/10/21...	U0000	NORMAL		000
EMPOT24:TSU07704	TSU07704			2008/10/20...	S622	ABENDed		
EMPOT241:JOB07570	JOB07570			2008/10/20...	U0004	NORMAL		004
EMPOT241:JOB04673	JOB04673			2008/10/10...	U0004	NORMAL		004
STEP1:JOB01329	JOB01329			2008/10/01...	U0000	NORMAL		000

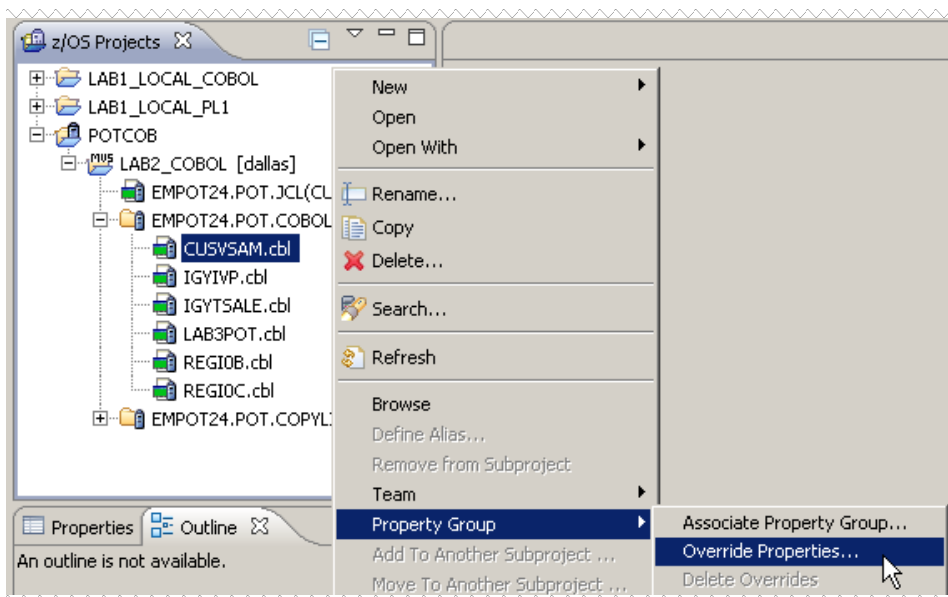
If you are not authorized to purge this job a message is displayed on the bottom:

Not authorized for job JOB07853

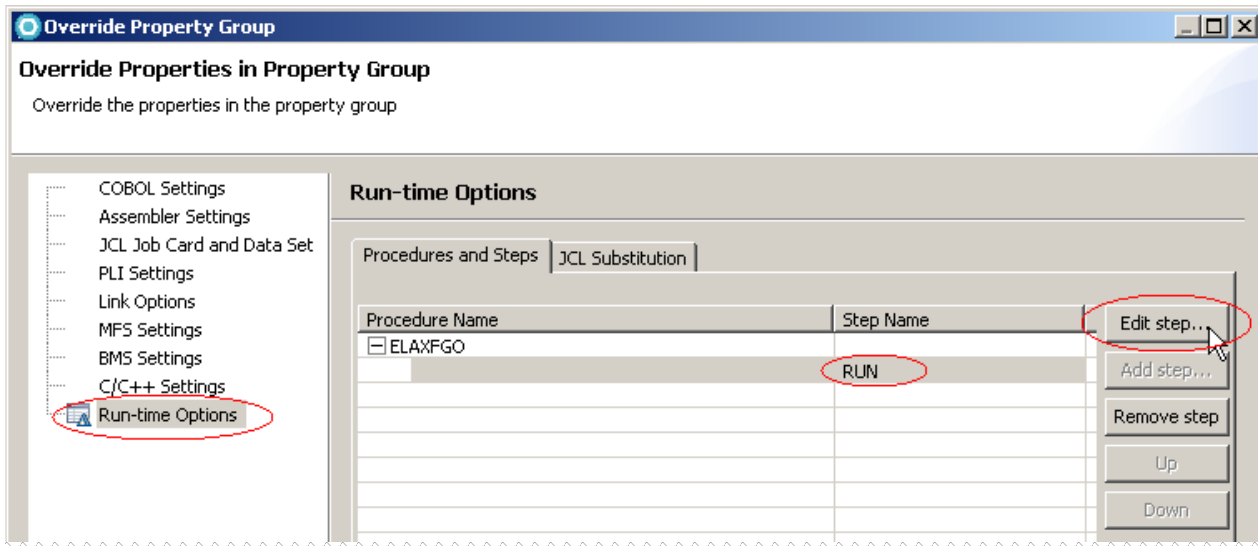
6.10 Generating JCL for z/OS Debug Execution

6.10.1 We need to change the Run-time properties of your project to be able to have the JCL generated with the debug options. We will override the default properties.

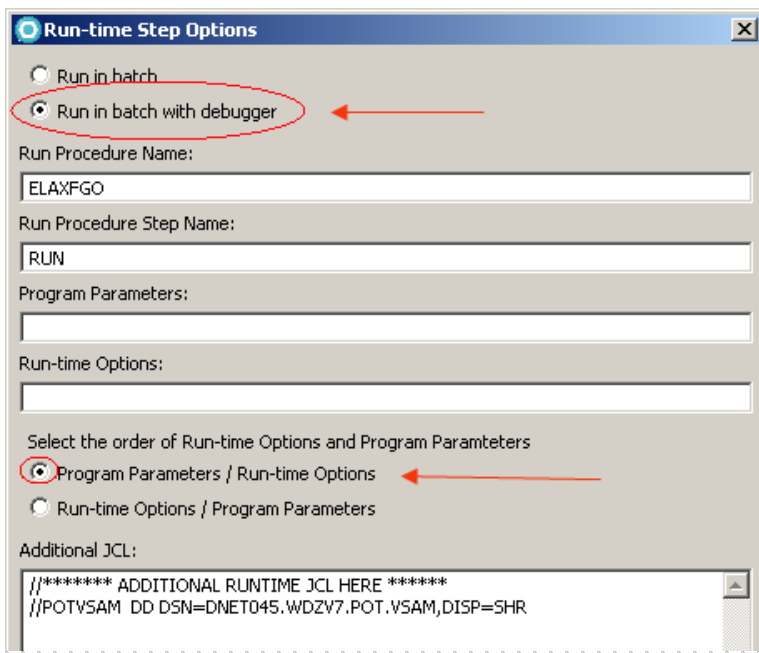
▶▶ Right click on **CUSVSAM.cbl** and select **Property Group → Override Properties**



6.10.2 **▶▶** Click on **Run-time Options**, expand **ELAXGO**, select **RUN** and click on **Edit step...**

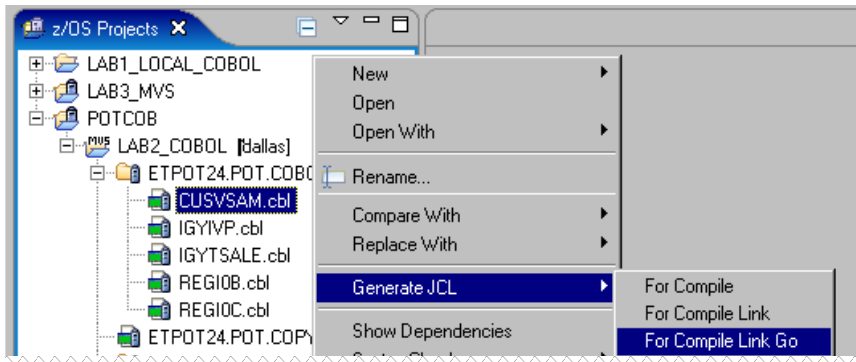


6.10.3 **▶▶** Change the option to **Run in batch with debugger** and be sure that **Program Parameters/Run-time Options** is selected and click **OK** and **Finish** to close the properties dialog:

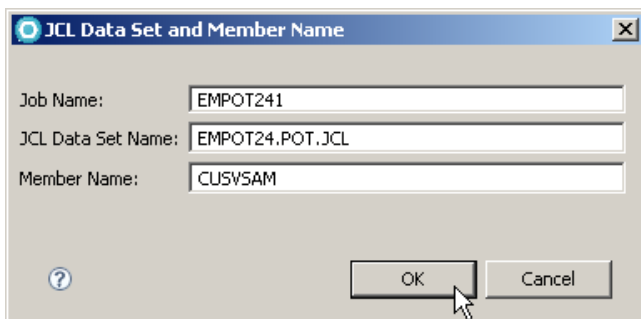


Note: the reason that we specified **Program Parameters/Runtime Options** is that we want that the JCL generated be in the form `PARM.RUN=('/TEST(,,TCPIP&&xx.xx.xx.xx%8003:*)'` to be used by the z/OS Debug. If you specify **Run-time Options/Program Parameters** the JCL generated will be in the form `PARM.RUN=('/TEST(,,TCPIP&&xx.xx.xx.xx%8003:*)/'` and our COBOL z/OS debug will not work this way. But when doing PL/I this is the preferred way.

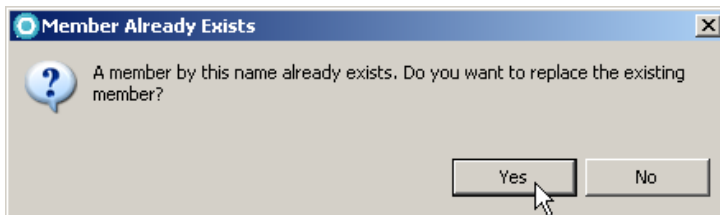
6.10.4 ▶▶ Right-click on **CUSVSAM.cbl** and select **Generate JCL → For Compile Link Go**.



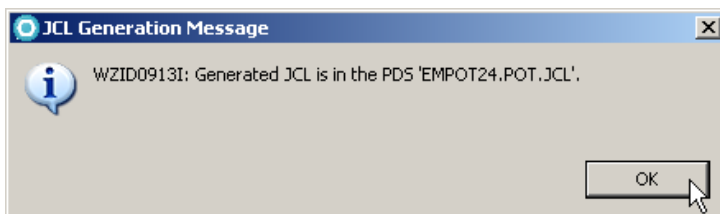
6.10.5 ▶▶ On the **JCL Data Set and Member Name** window, click **OK**.



6.10.6 ▶▶ Since this is the second time that you generate the JCL, click **Yes** to replace the existing member

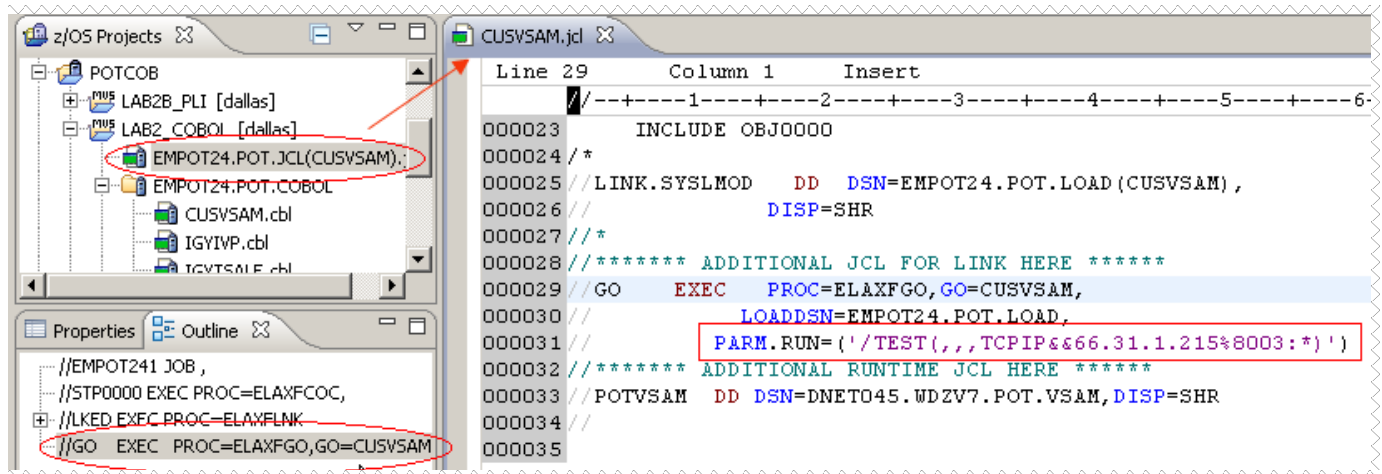


6.10.7 ▶▶ You should see the message below. Click **OK**.



6.10.8 **▶▶** Open the JCL file **double clicking** on it.

▶▶ Using the Outline view, click on **//GO** and note that a **PARM.RUN** parameter was generated. Note that the TCPIP address of your machine is also generated (and will be other than the one in the figure below). Your EM4Z workstation will be listening at the port **8003** (default is 8001 but we changed it on your workspace to 8003). The z/OS Debug tool running on the z/OS will communicate with you using this IP address (that is the why this section will work **ONLY** in some networks that are known by the z/OS in Dallas).



6.10.9 **▶▶** Close the editor pressing **Ctrl + F4**.

Do not submit this JOB unless you are instructed to do it.

What have you done so far?

At this point you are still connected to z/OS. In this Section 6 you edited a COBOL program located in Dallas, made small changes on it and after the changes were committed on z/OS, you returned to the previous version using the local history. This is a nice Rational Developer for System z feature that might be very helpful.



You added an error in the program, checked the syntax and corrected the error.

You generated the JCL necessary to execute the batch program and submitted it for execution.

Also, you have seen how the z/OS Debug tool would communicate back to your workstation using the TCP/IP address automatically generated in the JCL card.

If you are on the same network as the z/OS Dallas system you will be able to debug the code, the instructor will inform if this would be possible or not, otherwise (if you still have time), you might continue with section 7 that shows how to work offline.

6.11 Using the z/OS Debug Tool (Remote Debug)

ATTENTION **

Ask the instructor if you are able to do the z/OS debug exercise, otherwise you might see the debug on action using a provided flash Movie.

The z/OS Remote debug will work only if the z/OS Dallas system is able to understand your workstation's IP address.



When the COBOL program is running under z/OS Remote z/OS debug, it must be able to communicate back using the TCP/IP address that is known in the z/OS network. If you are behind a router or without an external physical TCP/IP address, the debug will not work. Usually this means that you must be inside the same network as the z/OS server.

But at least you have been able to generate the JCL and understand how the debug will work.

Also a flash movie will be provided for you to show you the debug.

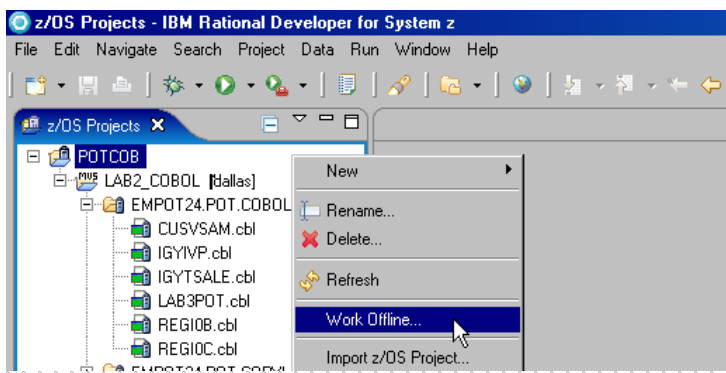
Section 7 – **Optional** - Working offline in a z/OS Project

We will see how to do some offline work.

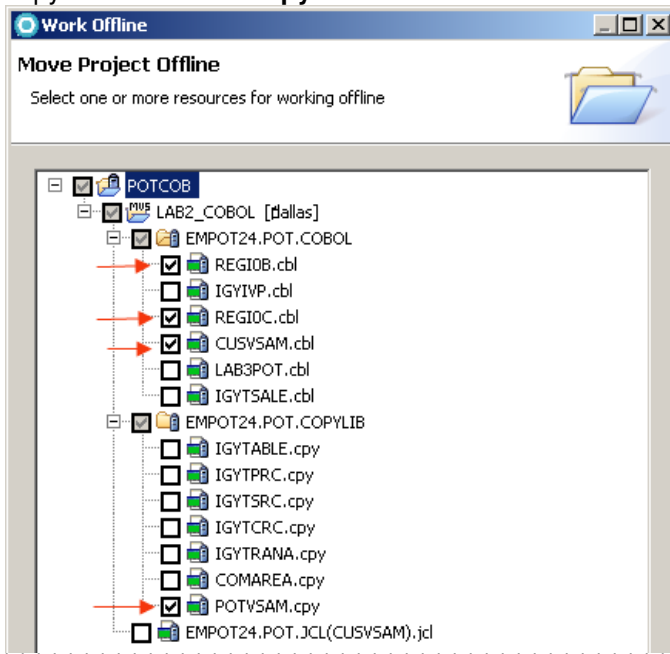
To create an MVS project, you have to be connected to a remote system. However, you do not have to remain connected to the remote system to work on resources associated with the project. You can work offline. Here we will explain how this can be done. **This example assumes that you are connected to a z/OS**

7.1 Using the **z/OS Projects** perspective and **z/OS Projects** view.

▶▶ Right-click in the MVS project named **POTCOB** that we created for the lab and select **Work Offline....**

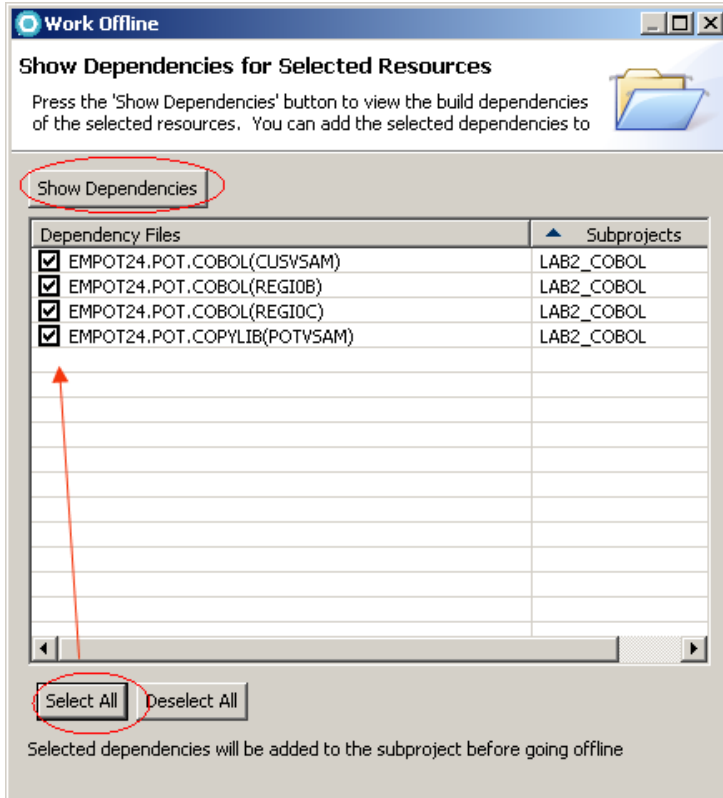


7.2 **▶▶** Expand **POTCOB** and select the data set members **REGI0B.cbl**, **REGI0C.cbl**, **CUSVSAM.cbl**, , and the copybook **POTVSAM.cpy** that we want to save to the workstation, to work without connection, and click **Next** .

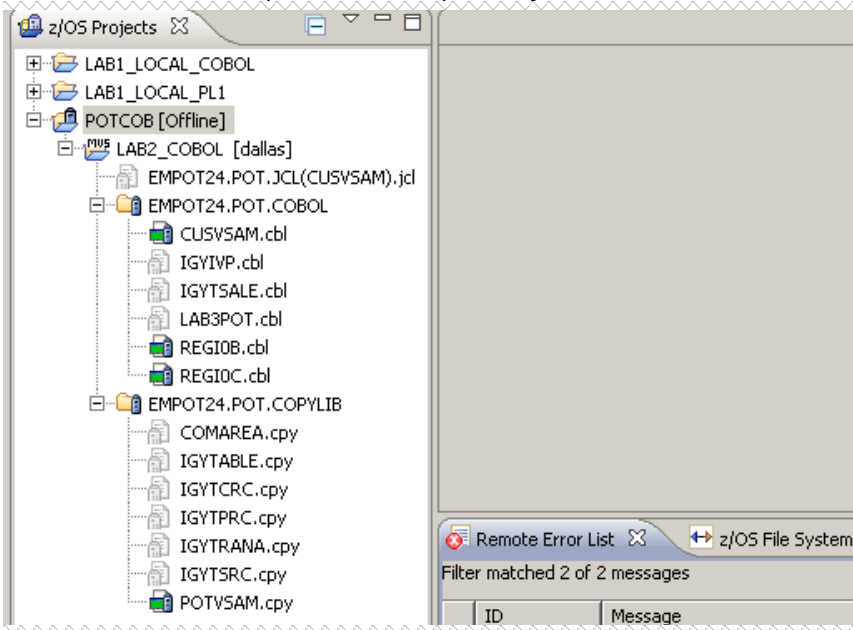


7.3 **▶▶** Click **Show Dependencies** button. This will submit a JOB to z/OS to find if some of the assets selected has some dependencies. (may take a while) and the result should be show as below.

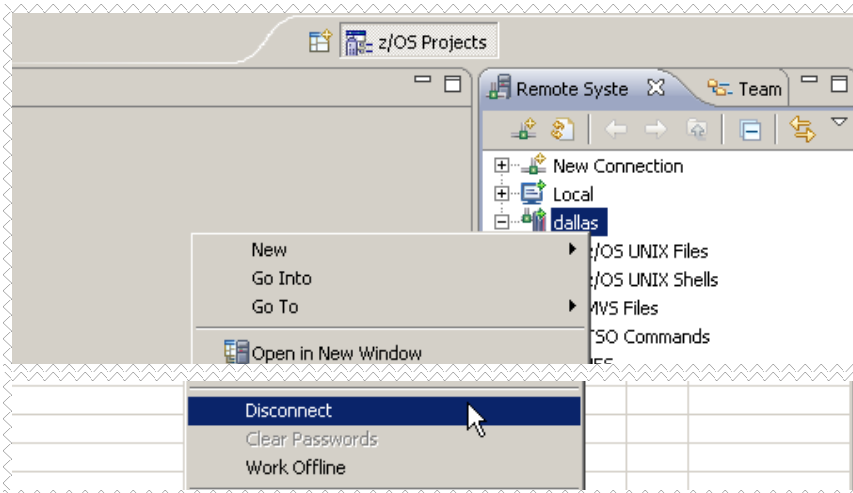
▶▶ Click **Select All** and **Finish**.



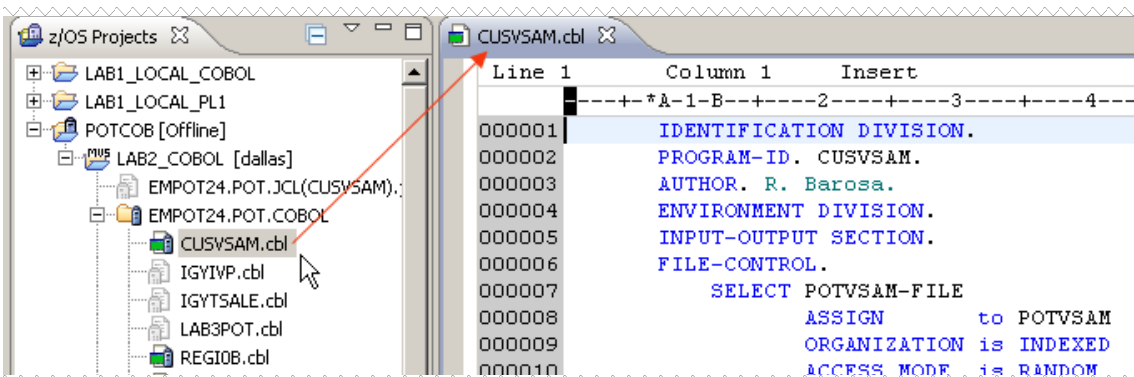
7.4 The selected components are copied to your local windows machine..



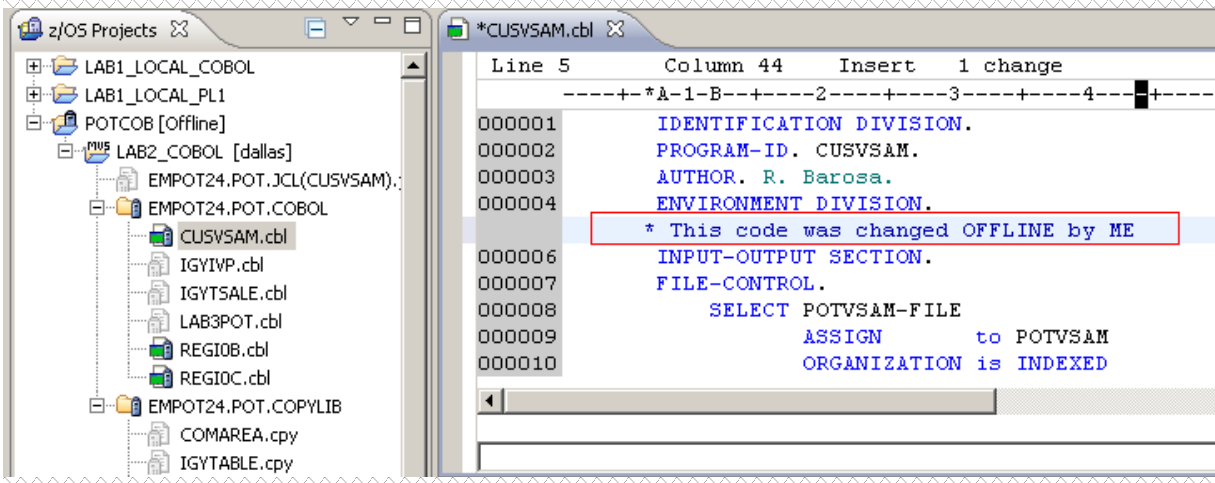
7.5 Using the **Remote Systems** view, **Right-click** on **dallas** click **disconnect** to be disconnected from the z/OS:



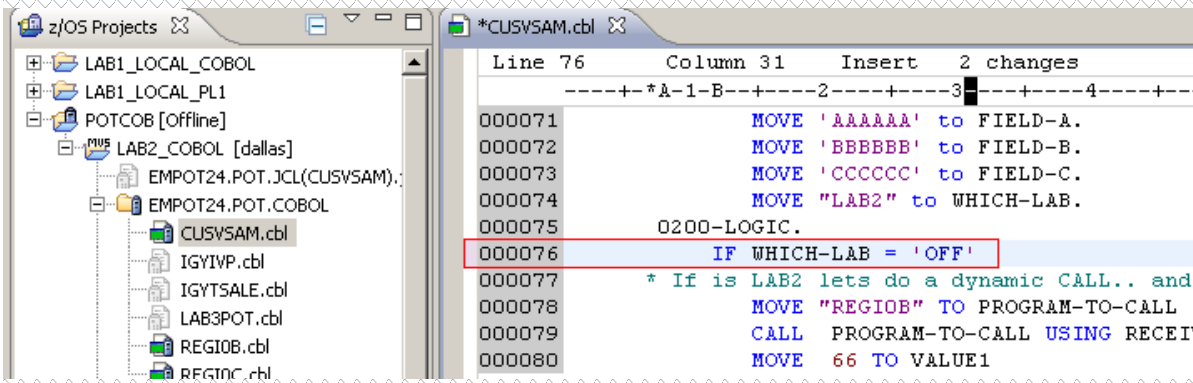
7.6 Switch back to the **z/OS Projects** view and edit **CUSVSAM.cbl** (double click on it). Now this program is copied in your workstation and you could work without z/OS connections.



7.7 **▶▶▶** Move the mouse before the number **000004** and left click, then press **CTRL + ENTER**, use the Tab key and enter a comment below the line 4 (Note that an * must be in the column 7).



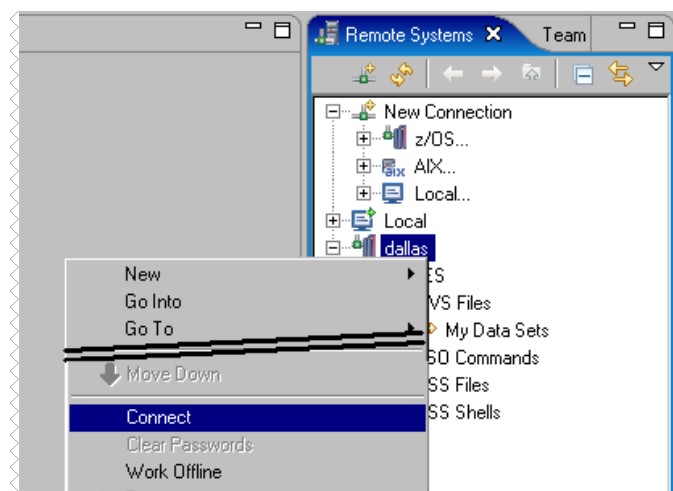
7.8 **▶▶▶** Change the statement **000076** as shown below (was 'LAB2' before, now it is 'OFF'):



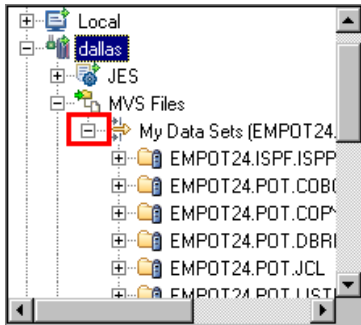
7.9 **▶▶▶** Save the code and **Close** the editor (**Ctrl + Shift + F4**). Click **Yes** to save the changes

7.10 **▶▶▶** To connect your MVS project back to the remote system, switch to the **Remote Systems** view and connect to z/OS.

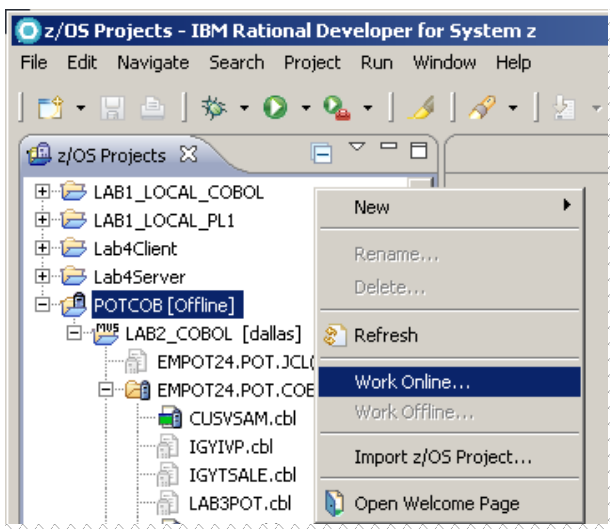
▶▶▶ **Right-click** on **dallas** and select **Connect**. The dallas icon will change to **dallas** :



7.11 **▶▶** Click **YES** for SSL warning and after connected, expand the **MVS Files** and **My Data Sets** folder and make sure that you have your MVS data sets listed:




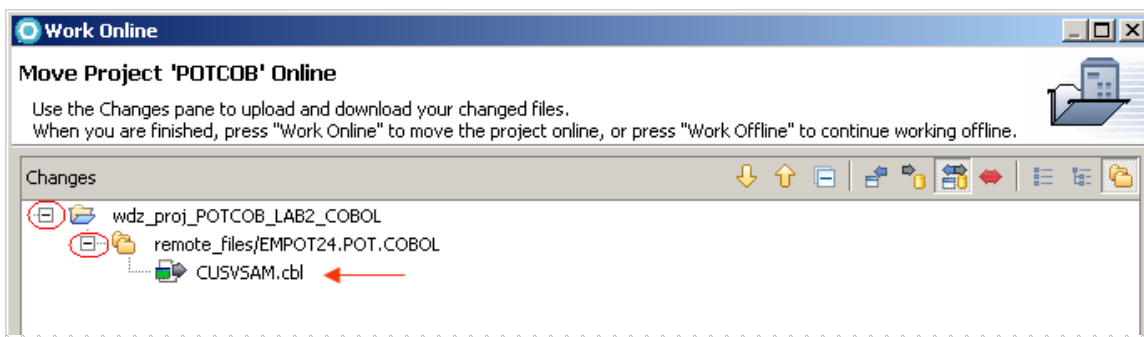
7.12 **▶▶** Switch back to the **z/OS Projects** View, **right-click** the **POTCOB** project and select **Work Online**.




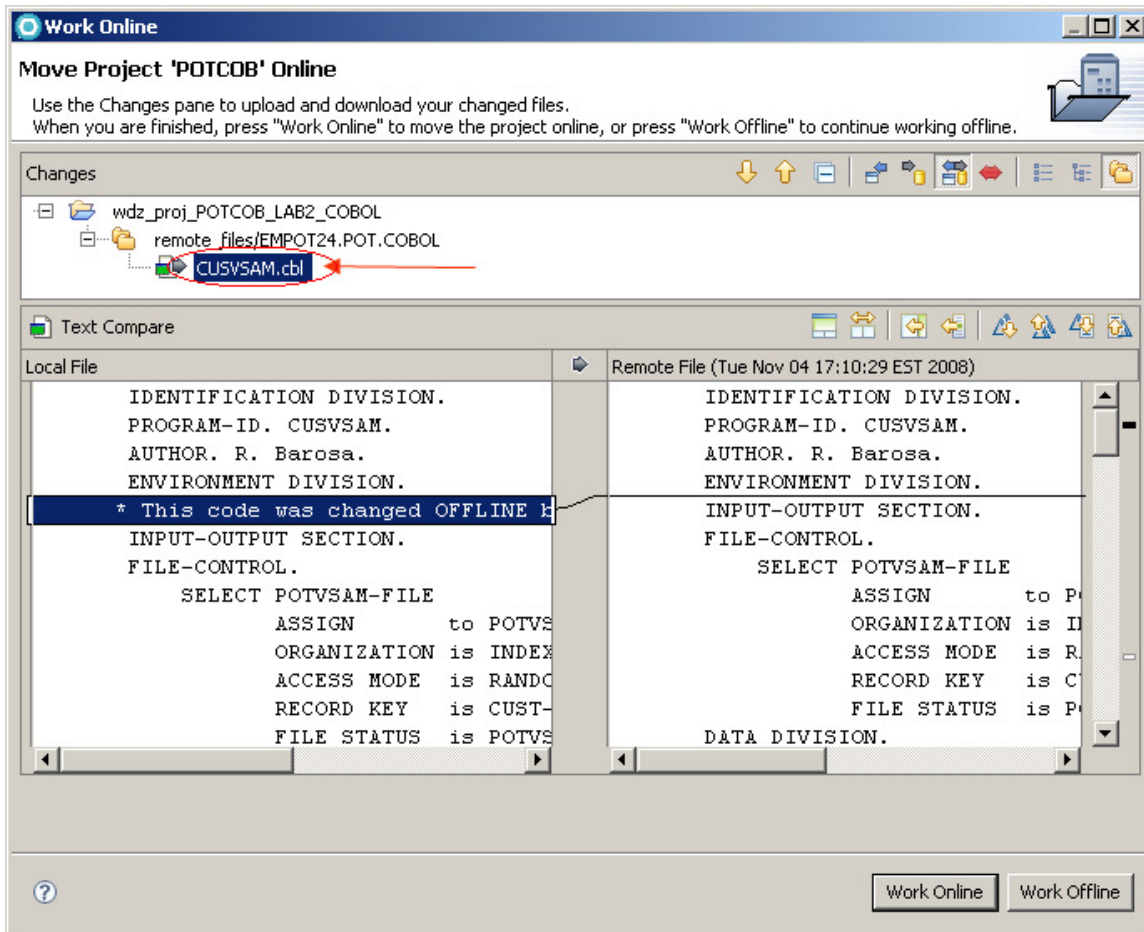
7.13 After synchronizing the differences you will have a screen as shown below.

▶▶ Expand **wdz_proj_POTCOB_LAB2_COBOL** and **remote_files/EMPOTXX.POT.COBO**

Note that **CUSVSAM.cbl** has an icon  that indicates that the CUSVSAM.cbl code must be uploaded to the host. Since we did not modify the other off-line assets they are not shown here.



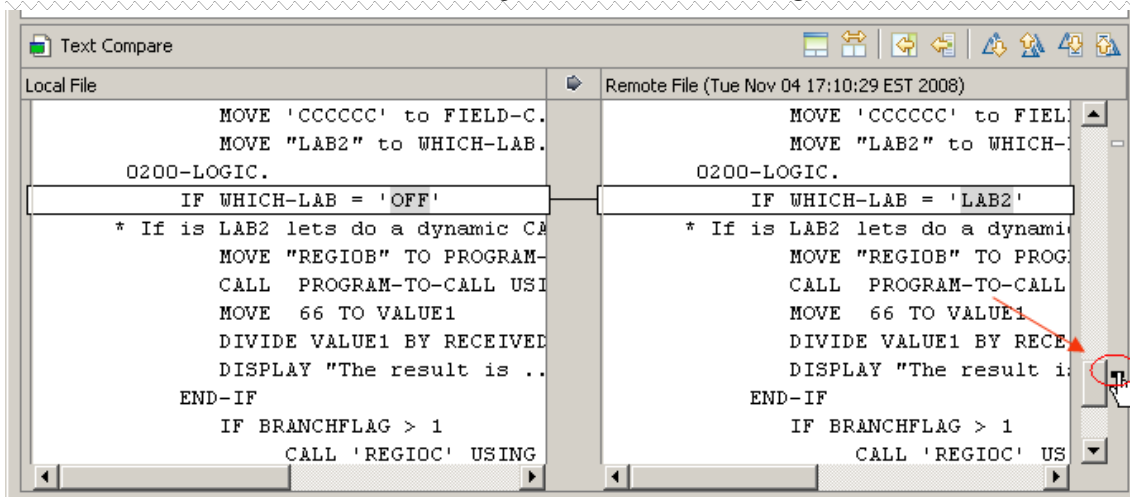
7.14  Click on **CUSVSAM.cbl** and you will see the changes that were made offline.



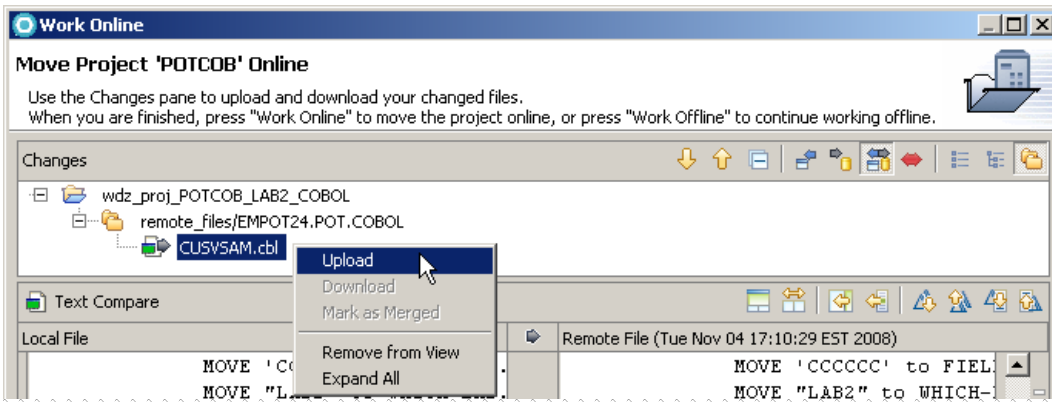
Note: You can also use these icons  to manage individual file changes.

And use these icons  to manage conflicts.

7.15  Click on the white mark on the right to see the **next change** as seen below:

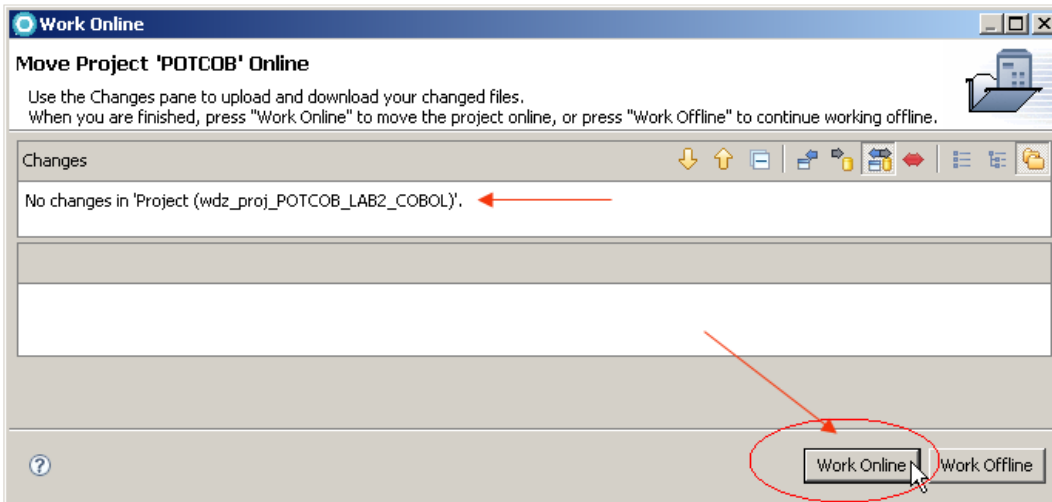


7.16 **▶▶** Right-click on **CUSVSAM.cbl** and select **Upload**:



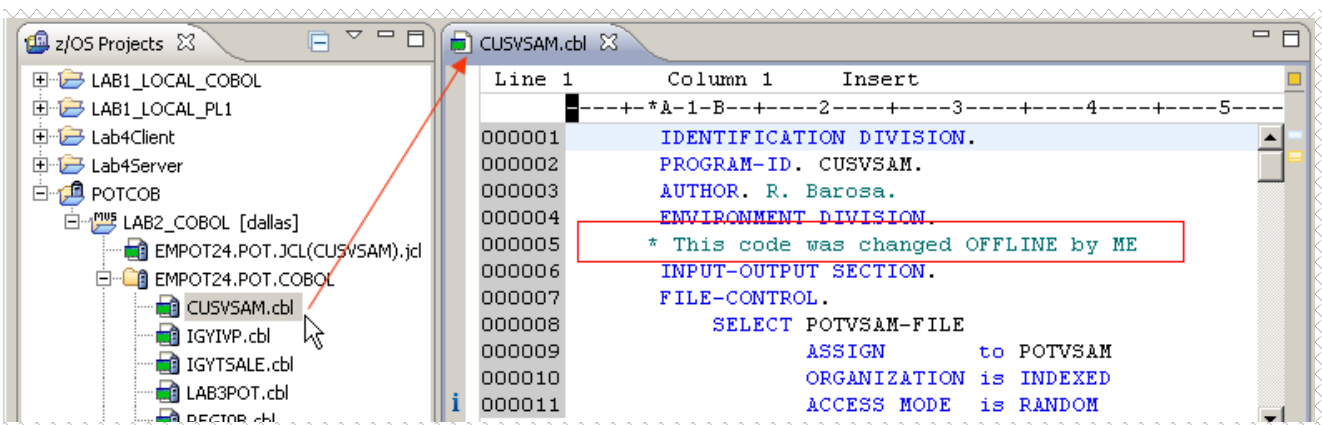
7.17 Note that there is no conflicts to solve, since no one has changed the z/OS host files during the offline work you have done. If the host files were modified you would have conflicts here.

▶▶ Click **Work Online**.



7.18 Now the host reflects the local changes and we are able to work online again with your z/OS project and the local changes that were uploaded.

▶▶ **Double click** on **CUSVSAM.cbl** and check the changes that you did while off-line





'Local Syntax Check' has encountered a problem. An internal error occurred during: 'Local Syntax Check' ?

This is a know issue that may happen if you are using an underscore (“_”) in the project name as we are. This will be fixed soon.. To bypass, close RDz and reopen it again.

7.19 **Close** the editor. (**CTRL + Shift + F4**)

7.20 If you have seen enough, **disconnect from z/OS** and **close Rational Developer for System z**.

Section 8 – **Optional** - Exploring TSO Commands

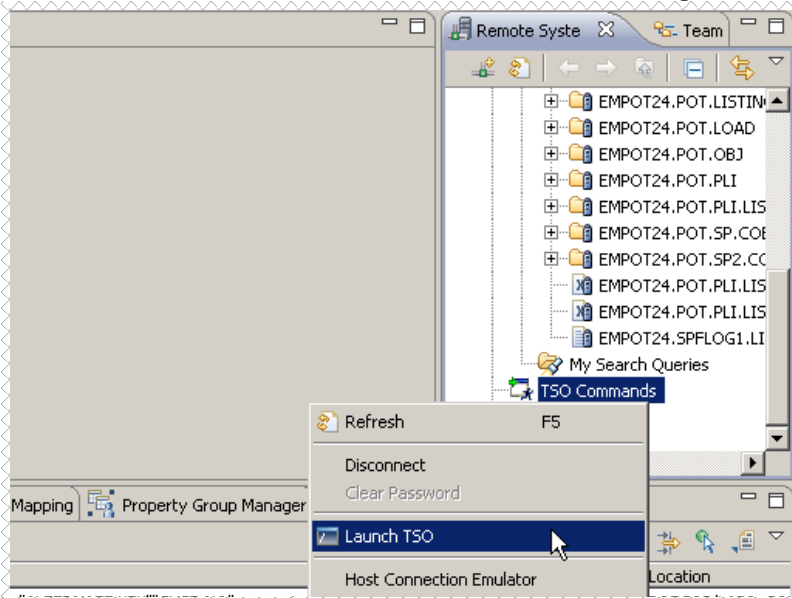
If you are interested, you can explore the **TSO Commands** feature of Rational Developer for System z

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 executed by RSE. Let's see one example below.

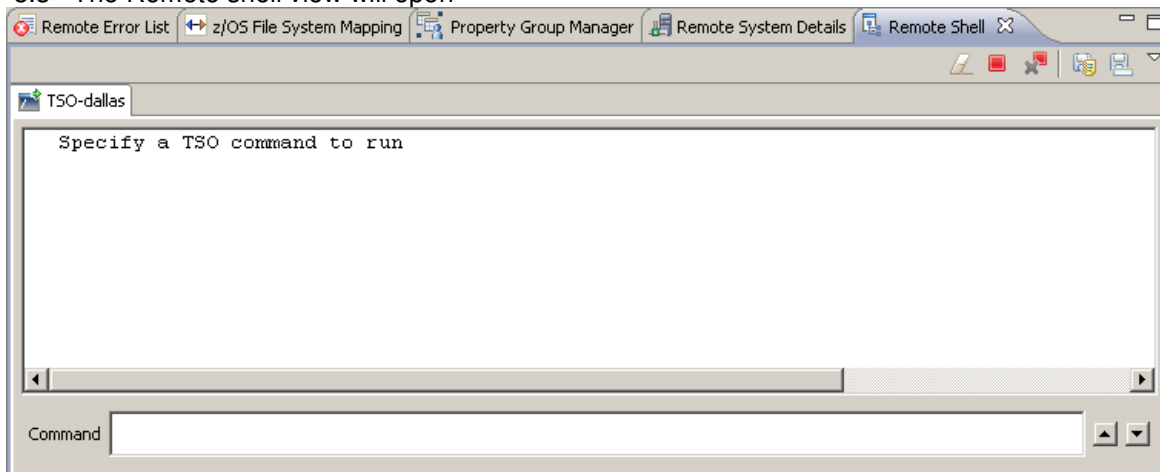
8.1 Using the z/OS projects view and Remote System perspective, be sure that you are connected to the z/OS. An easy way to verify is **left clicking** on the node **My Data Sets** to see your datasets on MVS.

If not connected, **right-click** on **dallas** and select **connect**.

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

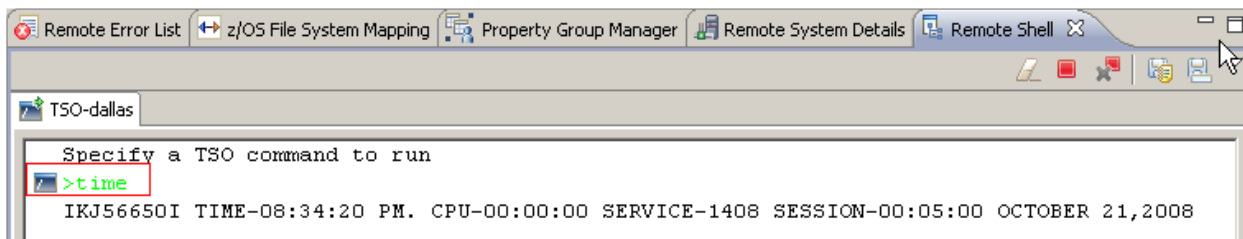


8.3 The Remote shell view will open

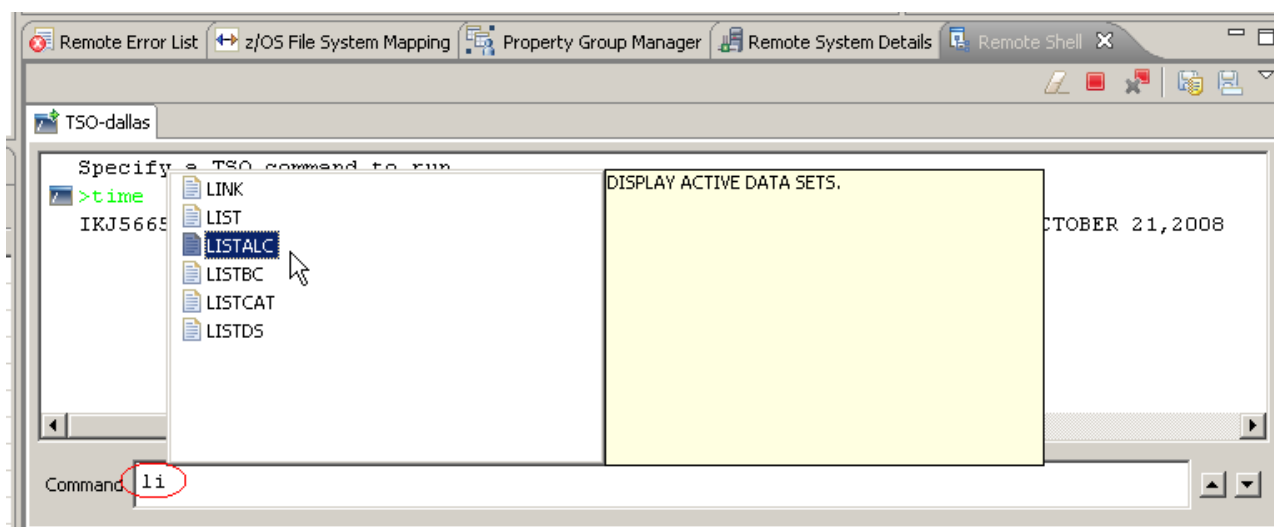


8.4 You will now be able to execute commands, like the command **time** seen below:

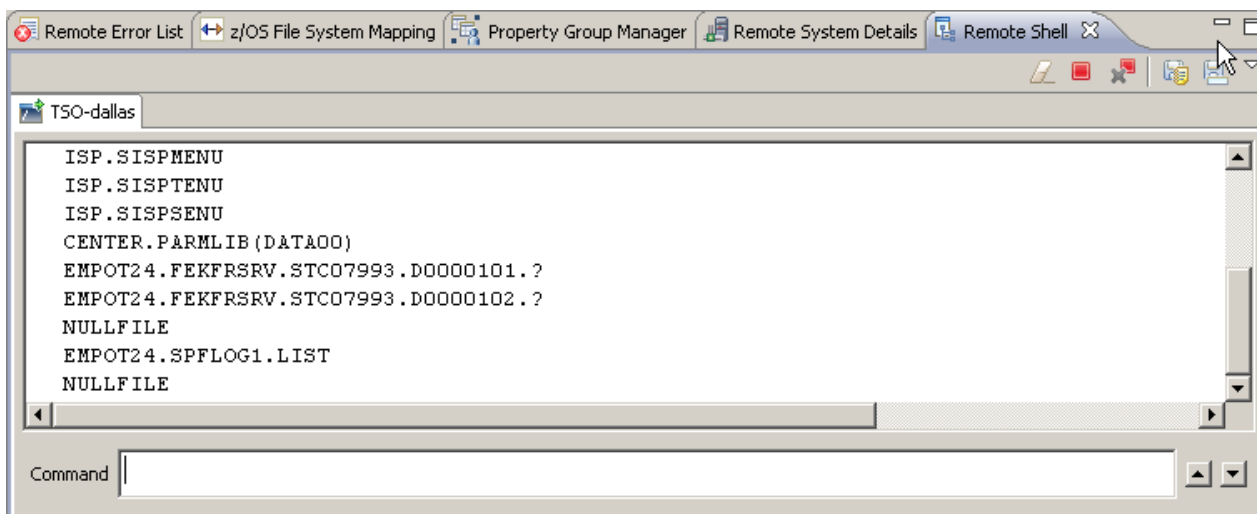
▶▶ Type **time** in the Command line and press **enter**.



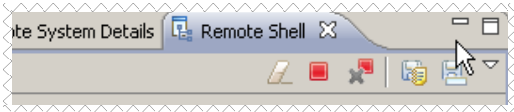
8.5 ▶▶ Also in the command line you will be able to use the Ctrl + Space to have the content assist, as seen below. Type **li** and press **Ctrl + Space**:



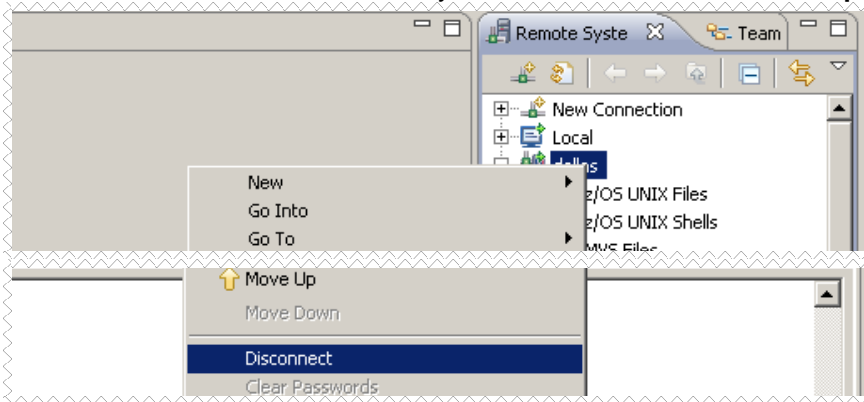
8.6 ▶▶ Select **LISTALC** and press **Enter**. The result will be the one below



8.7 You also could use the icons below and perform some activities..., Like Export the output, etc...

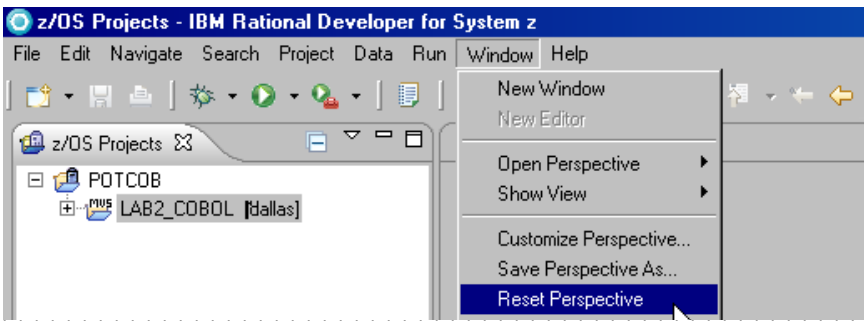


8.8 **Disconnect** from the z/OS system and **close Rational Developer for System z.**

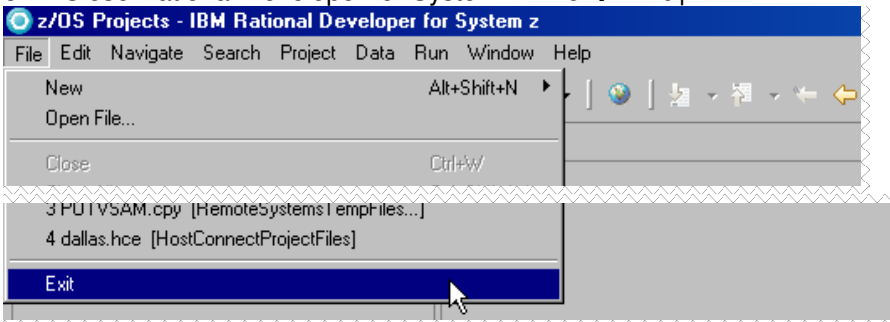


8.9 **Close** all open editor windows if any opened... (pressing **Ctrl+Shift+F4** should accomplish this).

8.10 **Reset** the perspective to the defaults. Select **Windows → Reset Perspective**



8.11 **Close** Rational Developer for System z: **File → Exit**





Congratulations! You have completed the Lab. .

Solution

If you could not complete the lab don't get frustrated. In case you missed one step or typed a wrong name you would have problems. In that case you can use the solution workspace.

This workspace is located at **C:\Workspaces7.5\RDZSOLUTION**

To start the workspace with the solution:

▶▶ Open the folder  and double-click in the icon  RDz7 solutions
Or specify this workspace when Rational Developer for System z starts:

