

# Rational Developer for i Sandbox for IBM i Lab Exercise Workbook

## Rational Developer for i

### Lab 04 - Debug

This lab shows you how to debug a CL/RPG program.

Version 6, January 2021

The most up to date version of this document can be found on Rational Developer for i - Hands-On Labs at [http://ibm.biz/rdi\\_labs](http://ibm.biz/rdi_labs).



© Copyright International Business Machines Corporation, 2021. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

LAB 04 - DEBUG .....	4
Overview .....	4
Learning objectives .....	4
Skill level and prerequisites .....	4
Conventions used in this workbook .....	5
Client System requirements .....	5
Host System requirements .....	5
1    INTRODUCING THE INTEGRATED IBM I DEBUGGER .....	5
2    STARTING A DEBUG SESSION USING A SERVICE ENTRY POINT .....	7
2.1 Changing the Update Production File preference .....	7
2.2 Set Service Entry breakpoint .....	8
2.3 Setting breakpoints .....	14
2.4 Adding a conditional breakpoint .....	15
2.5 Monitoring variables .....	18
2.6 Error handling .....	20
2.7 Stepping into a program .....	23
2.8 Listing call stack entries .....	24
2.9 Setting breakpoints in PAYROLLG .....	25
2.10 Removing a breakpoint in PAYROLLG .....	27
2.11 Monitoring variables in PAYYROLLG .....	29
2.12 Adding a memory monitor .....	31
2.13 Setting Watch breakpoints .....	33
2.14 Terminate a debug session .....	35
3    DEBUGGING A JOB .....	37
4    LAB SUMMARY .....	39
CONGRATULATIONS! .....	40
APPENDIX A    NOTICES .....	41
APPENDIX B    TRADEMARKS AND COPYRIGHTS .....	43

## Lab 04 - Debug

### Overview

This module teaches you how to debug a CL and ILE RPG program.

### Learning objectives

You will learn how to start the debugger, set breakpoints, monitor variables, run, and step into a program, view the call stack in the Debug view, remove a breakpoint, add a memory monitor, and set Watch breakpoints and all from the Debug perspective.

### Skill level and prerequisites

Introductory.

#### Important!



You should complete **RD*i* Lab01** 'Getting started' before you work on this lab. Lab01 contains the following information and instructions:




- Which IBM i server to connect to
- Which User ID to use
- How to start RD*i*, create a connection and connect
- How to setup the correct library list for this lab

Knowledge of basic Microsoft Windows operations such as working with the desktop, mouse operations such as opening folders and drag-and-drop is assumed. It will also be helpful if you understand DDS and ILE RPG.

## Conventions used in this workbook

<b>Bold font</b>	is used to highlight user interface controls
Mono-spaced font	is used for user input text and code blocks
<i>Italic font</i>	is used for variable names and glossary terms

The following icons are also used to identify categories of information:

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step but is helpful or good to know.
	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

## Client System requirements

The labs require IBM Rational Developer for IBM i (RDi) to be installed on your workstation. If you do not yet have this, you can download it for free from [http://ibm.biz/rdi\\_trial](http://ibm.biz/rdi_trial).

## Host System requirements

The easiest way to ensure you have everything you need, is to use the demonstration IBM i server that is set up and ready to use with these lab exercises. These labs use the RSELABxx library on that system. For those who want to run these labs on to their own system, you can load a SAVF with the RSELABxx library from [rselabxx.savf](http://rselabxx.savf).

# 1 Introducing the Integrated IBM i debugger

The Integrated IBM i Debugger is a source-level debugger that enables you to debug and test an application that is running on an IBM i system. It provides a functionally rich interactive graphical interface that allows you to:

- View source code or compiler listings, while the program is running on an IBM i host.
- Set, change, delete, enable and disable line breakpoints in the application program. You can easily manage all your breakpoints using the Breakpoints view.
- Set Watch breakpoints to make the program stop whenever a specified variable changes.
- View the call stack of your program in the Debug view. As you debug, the call stack gets updated dynamically. You can view the source of any debug program by clicking on its call stack entry.
- Step through your code one line at a time.
- Step return, step into or step over program calls and ILE procedure calls.
- Suspend program execution and get control back to the debug session.
- Display a variable and its value in the Monitors view. The value can easily be changed to see the effect on the program's flow.
- Locate procedure calls in a large program quickly and easily using the Modules/ Programs view.
- Debug multi-threaded applications, maintaining separate stacks for each thread with the ability to enable and disable any individual thread.
- Load source from the workstation or a different IBM i system than the one the program runs on – useful if you don't want the source code on a production machine.
- Debug client/server and distributed applications.

The Debugger supports RPG/400® and ILE RPG, COBOL and ILE COBOL, C, C++ and CL.

Now that you know the basic features of the debugger, let's try them out.

## 2 Starting a debug session using a service entry point

You will be working with the ILE RPG program PAYROLLG.

The instructions for creating a connection to the IBM i system are contained in Lab01 of this series of Labs. If you haven't worked through Lab01, please do this first. The instruction in this Lab depends on the correct setup of a connection to an IBM i server.



**Note:**

PAYROLLG is the same RPG program as PAYROLL but without compile errors. You are using it instead of PAYROLL in this lesson, to accommodate anyone who decided to skip right to this Lab, without completing Lab02 or Lab03.

To make the lesson more interesting you will use CL program **CLR1** to call **PAYROLLG** and you will pass one parameter to CLR1.

In this lesson, you will use a service entry point to start a debug session for your application. The Service Entry Point feature is designed to allow easy debugging of applications on IBM i that invoke business logic written in RPG, COBOL, CL, C, or C++. The service entry point is a special kind of entry breakpoint that can be set directly from the Remote Systems Explorer. It is triggered when the first line of a specified procedure is executed in a job that is not under debug. Service entry points allow you to gain control of your job at that point. A new debug session gets started and execution is stopped at that location.

### 2.1 Changing the Update Production File preference



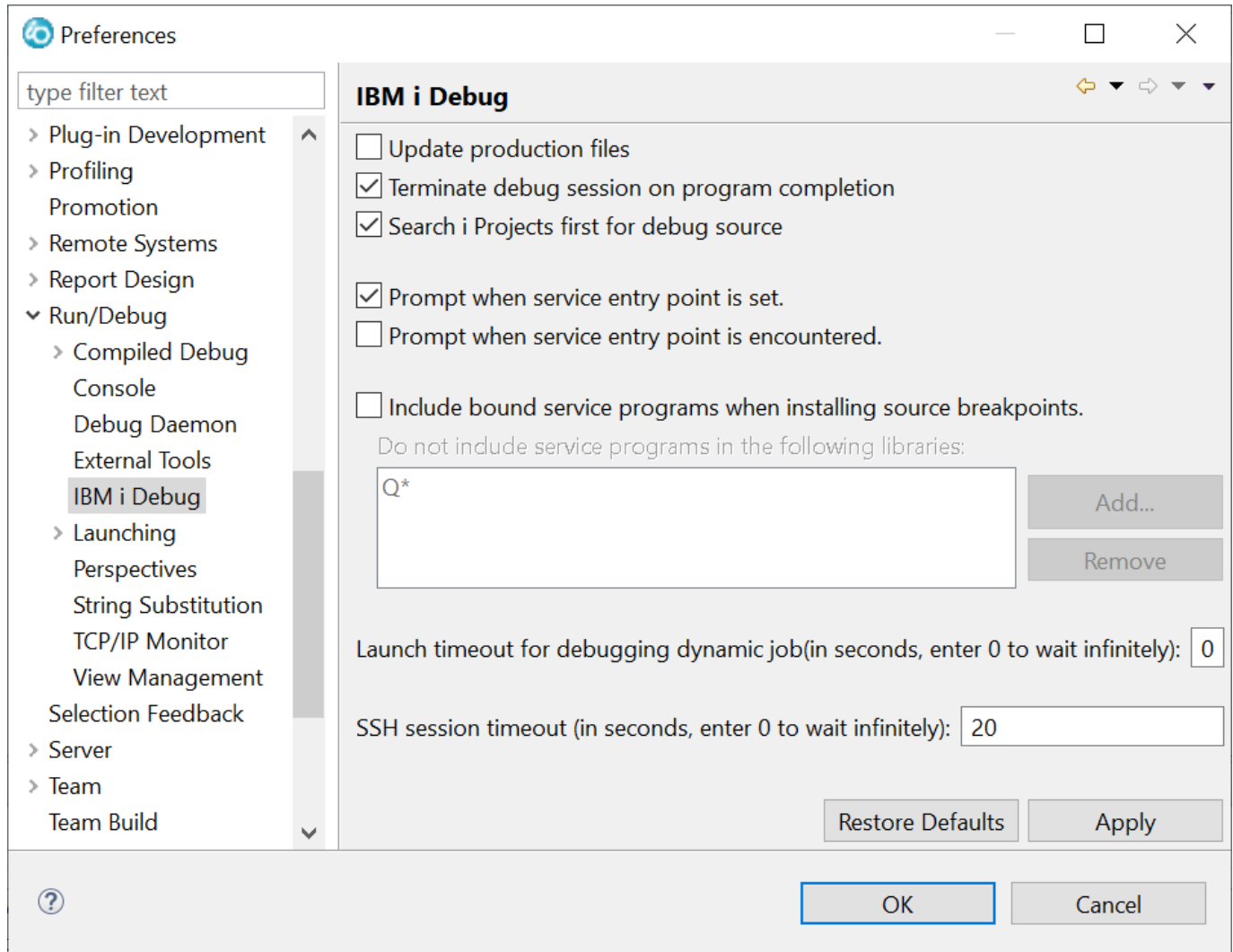
**Note: This section is for your information only, continue to the next section to start the exercises.**

We first describe how to change the debug preference to allow update of Production libraries during a debug session. This preference is by default set to **not** allow update of production libraries.

For this exercise you don't have to change this preference but here are the steps to follow when you work with production libraries in your own environment.

To set the i Debug preference in the workbench:

- \_\_1. Click **Window > Preferences**
- \_\_2. Expand **Run/Debug**
- \_\_3. Select **IBM i Debug**



Since you are using test libraries for these exercises, you don't have to change this IBM i Debug preference to "Update production files".

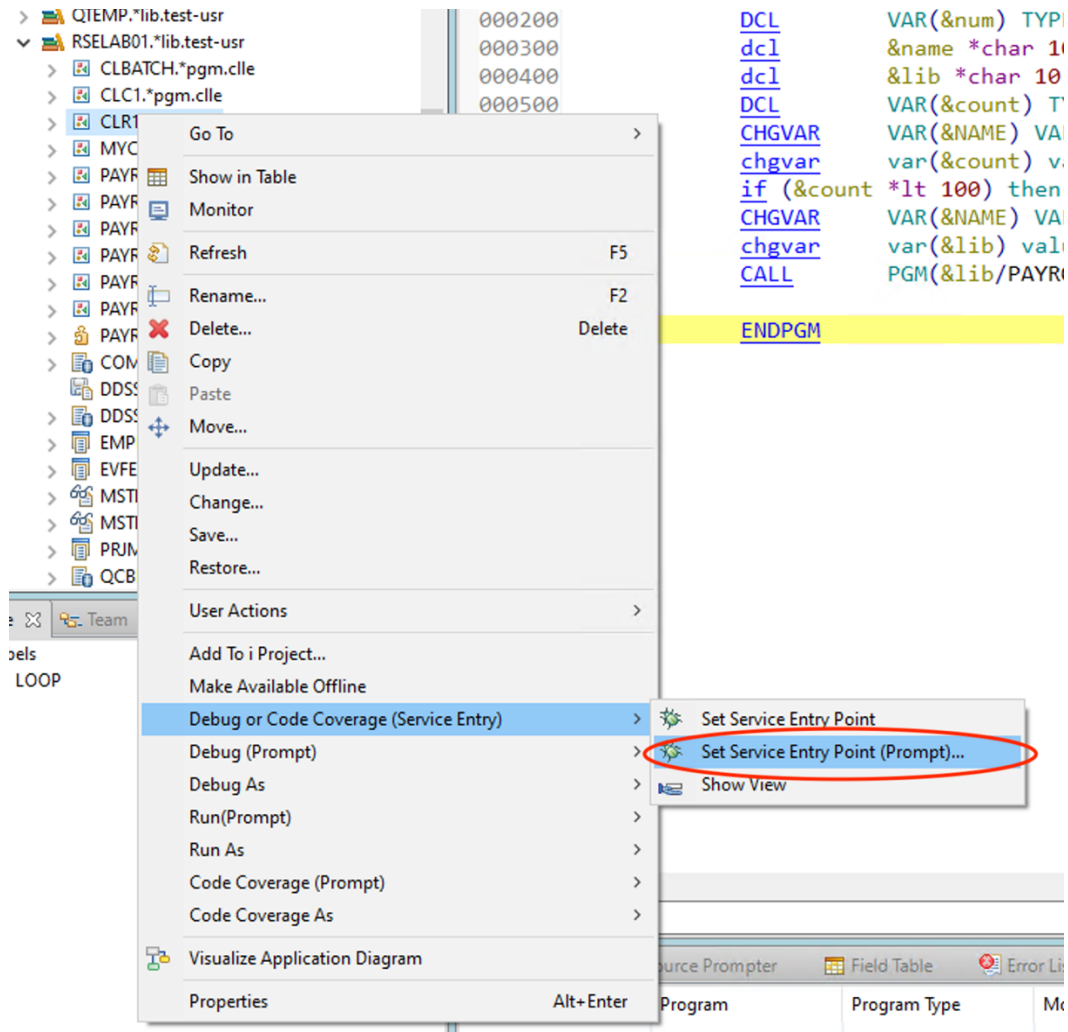
## 2.2 Set Service Entry breakpoint

Now let's begin with starting the debug session using a service entry breakpoint:

1. In the Remote Systems view expand the **Library list** filter, if it isn't expanded already.
2. Expand library RSELABxx, if it isn't expanded already.



3. Right-click program **CLR1** in library **RSELABxx**.



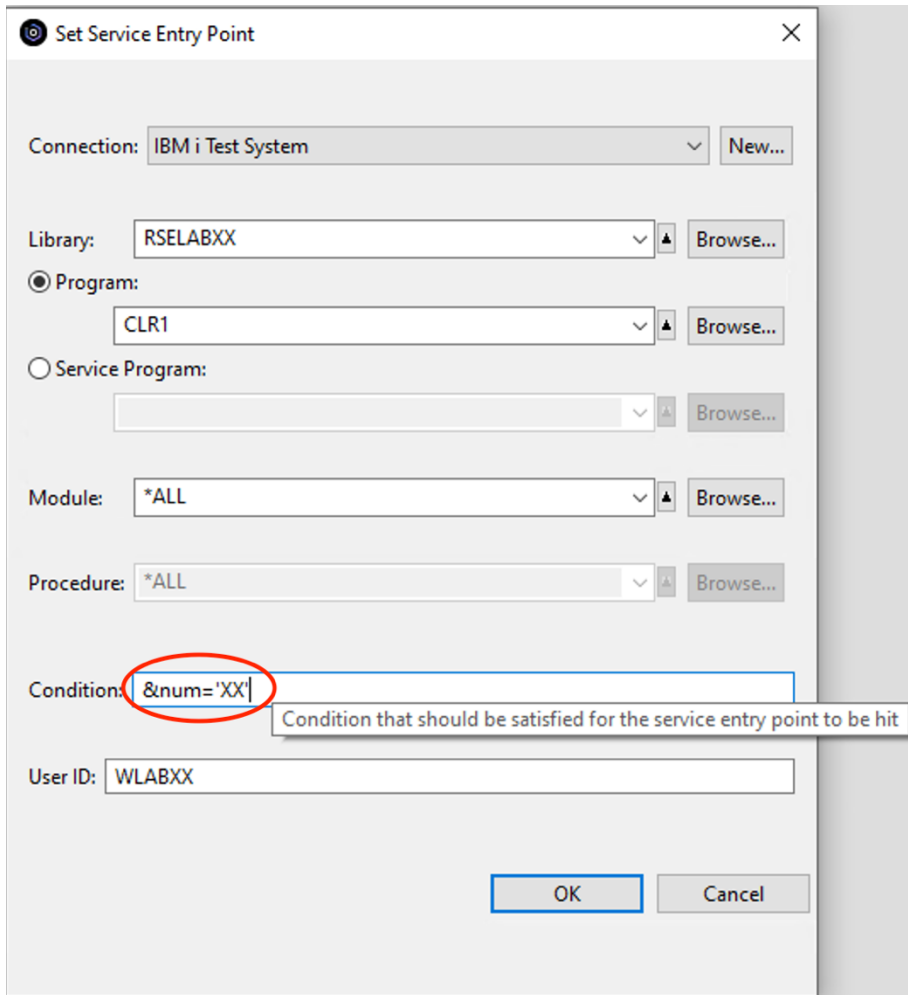
4. Click **Debug or Code Coverage (Service Entry) > Set Service Entry Point (Prompt)** on the pop-up menu to set a service entry point.



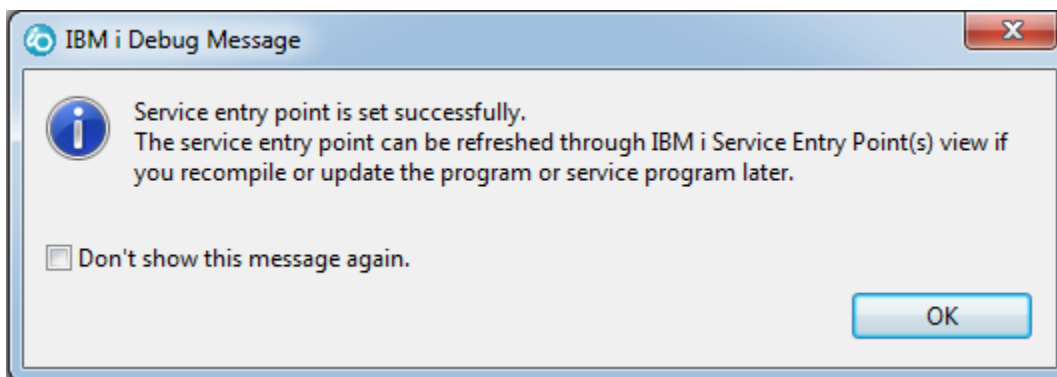
**Note:**

If you selected the unprompted menu item, the service entry breakpoint would be triggered, and you would be placed in the debugger whenever the CLR1 program is called no matter the value of the parameter.

We are going to specify a condition so that we are only placed into the debug when **&num** is equal to our team number. (make sure the **&num** is set to your two-digit team number corresponding to your **RSELABXX** library)

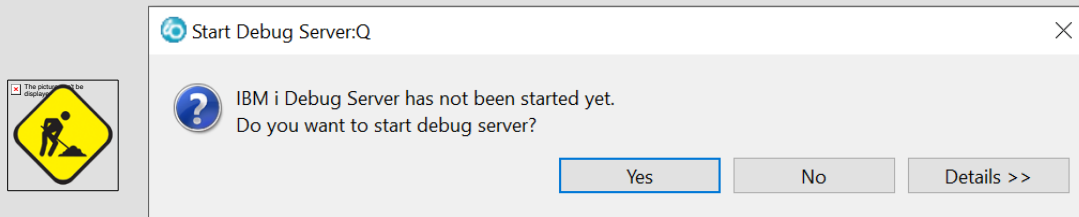


A message displays indicating the service entry point was successfully set.



## Troubleshooting

If you get this error message instead,

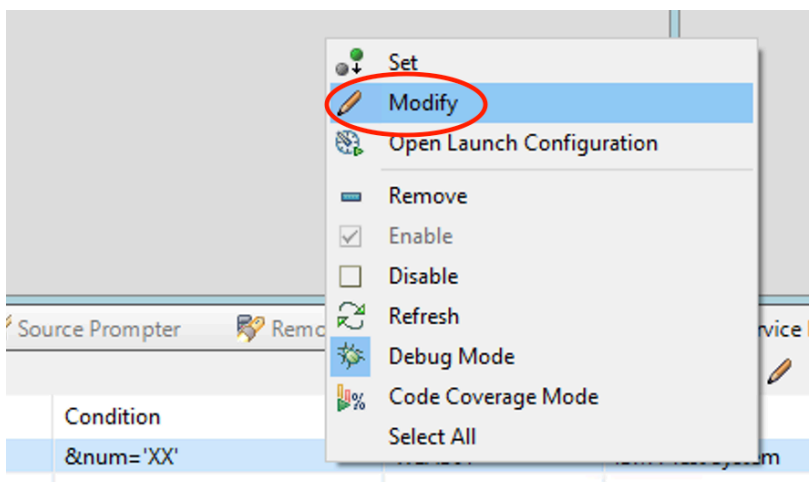


indicating that the Debug Server has not been started yet. Click Yes button to start the Debug server.

The **Service Entry Points** view is automatically added to the stacked views below the edit view. It lists all service entry points set in this workbench instance. You use this view to delete, activate, de-activate, modify and refresh service entry points.


Program	Program Type	Module	Procedure	Condition
CLR1	*PGM	*ALL	*ALL	&num='XX'

5. Note you can see the condition you specified. To modify the condition, right-click the condition you want to modify and click 'modify'.



\_\_6. For our purposes, we do not want to change the condition, so click 'Cancel'.

\_\_7. Switch to a 5250 emulation session.



**Troubleshooting**

If you have done the exercises in Lab03, your 5250 session might still be associated with the RSE server job and you will need to release the interactive session. To do so, in the Remote Systems view, right-click Objects and click **Release Interactive Job** on the pop-up menu.

On the command line of the 5250 screen, add the library RSELABxx to the library list and invoke the program CLR1, by entering the following commands:

- \_\_8. **ADDLIBLE RSELABxx** (with xx your team number).

```

11. iSeries Access tasks

90. Sign off

Selection or command
==> addlible rselabxx

```

F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assista

- \_\_9. First, we will call the program with a parameter other than your team number so that it will not trigger the SEP breakpoint. Enter **CALL PGM(RSELABxx/CLR1) PARM('AA')** (with xx your team number)

```

Selection or command
==> CALL PGM(RSELABXX/CLR1) PARM('AA')

```

- \_\_10. Note that we did not hit the debugger because the SEP condition is not fulfilled, hit F3 to quit.

```

Display Program Messages

Job 503331/WLAB01/QPADEV005X started on 01/20/21 at 11:31:19 in subsystem QI
CPF0001 received by procedure CLR1. (C D I R)

Type reply, press Enter.
Reply . . .

F3=Exit F12=Cancel

MA a MW 20/018

```

Notice the condition on the SEP allows us to avoid being put into the debugger for a different team number than our own.

- \_\_11. Now, let's call the program with your team number so that the SEP condition will be triggered. Enter **CALL PGM(RSELABxx/CLR1) PARM('xx')** (where xx is your team number).

```

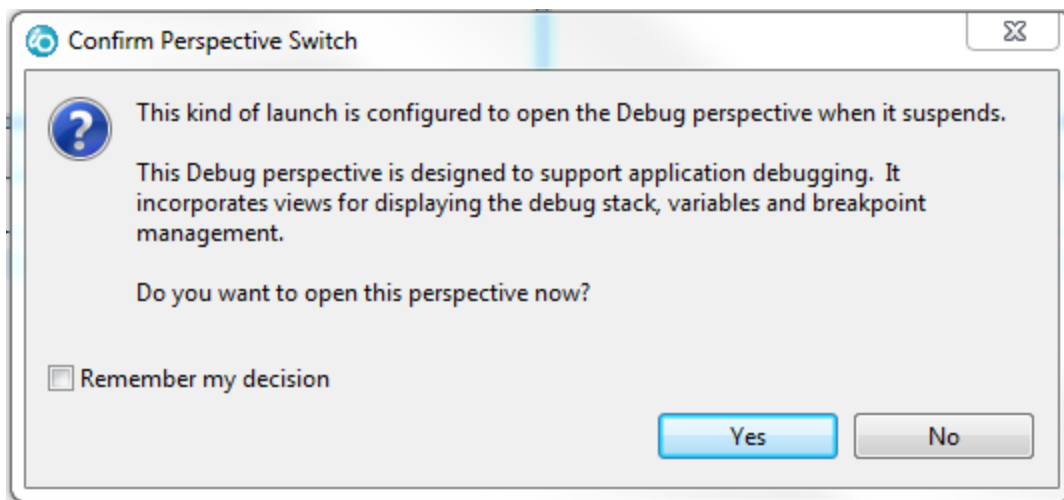
90. Sign off

Selection or command
==> CALL PGM(RSELABXX/CLR1) PARM('xx')

F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel  F13=Information Assistant

```

- \_\_12. A Confirm Perspective Switch dialog may appear:



If so, confirm by clicking **Yes**.

As soon as the program enters the system, the service entry point is hit, and the debug session is started on the workstation and the perspective displays with the CLR1 source code in the editor. The Debug perspective gives you access to all available debugger features. Let's look at some of them.

- \_\_13. Click anywhere in the workbench to give it focus.

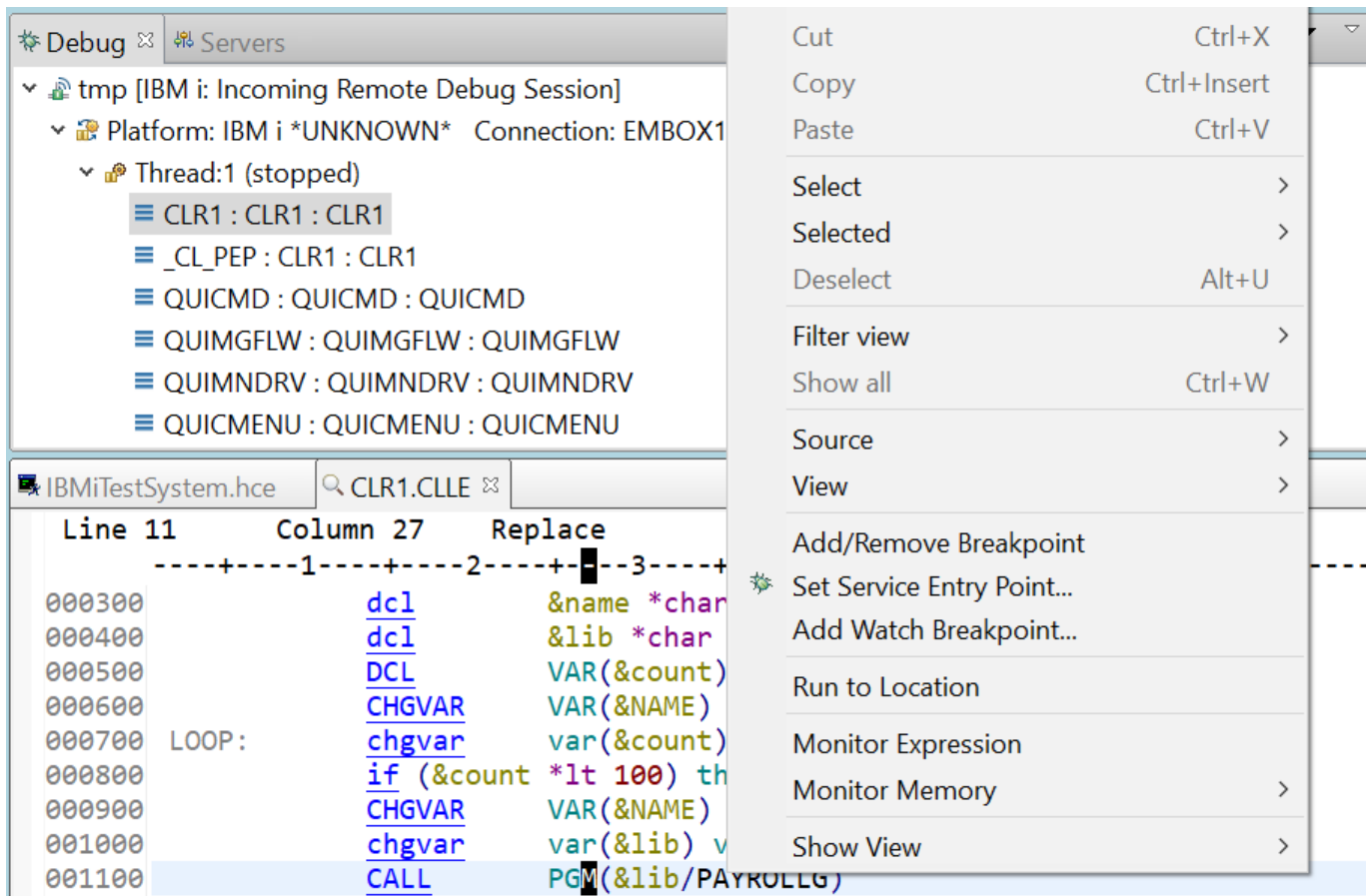
## 2.3 Setting breakpoints

You can only set breakpoints at executable lines. One way to set a breakpoint is to right-click on the line in the Source view.

To set a breakpoint:

- \_\_1. Position the cursor on line 11.
- \_\_2. Right-click anywhere on line 11.

- \_\_3. Click **Add/Remove Breakpoint** on the pop-up menu.



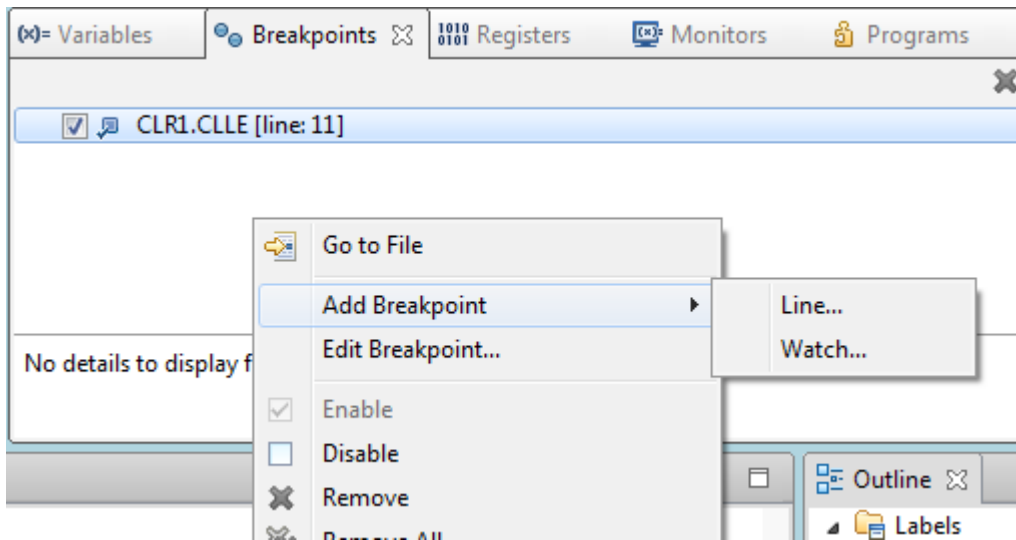
A dot with a check mark in the **prefix** area indicates that a breakpoint has been set for that line. The prefix area is the small grey margin to the left of the source lines.

Now you add a conditional breakpoint to stop in the loop when it loops the 99th time.

## 2.4 Adding a conditional breakpoint

- \_\_1. Select line 8.
- \_\_2. Click the **Breakpoints** tab in the upper right pane of the Debug perspective. The Breakpoints view opens.
- \_\_3. Right-click anywhere within the **Breakpoints** view.

4. Click **Add Breakpoint > Line** on the pop-up menu.



The Add a Line Breakpoint window opens.



\_\_5. Click **Next**.

**Add a Line Breakpoint**

**Required information**  
Sets a breakpoint to stop execution at a specific source line

Defer breakpoint until executable is loaded

Program  
\*PGM RSELAB01/CLR1

Module  
CLR1

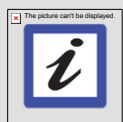
Views  
 \*SOURCE  
 \*LISTING  
 \*STATEMENT

Source(optional):  
CLR1.CLLE

Line:  
8

User label (optional):

? < Back Next > Finish Cancel



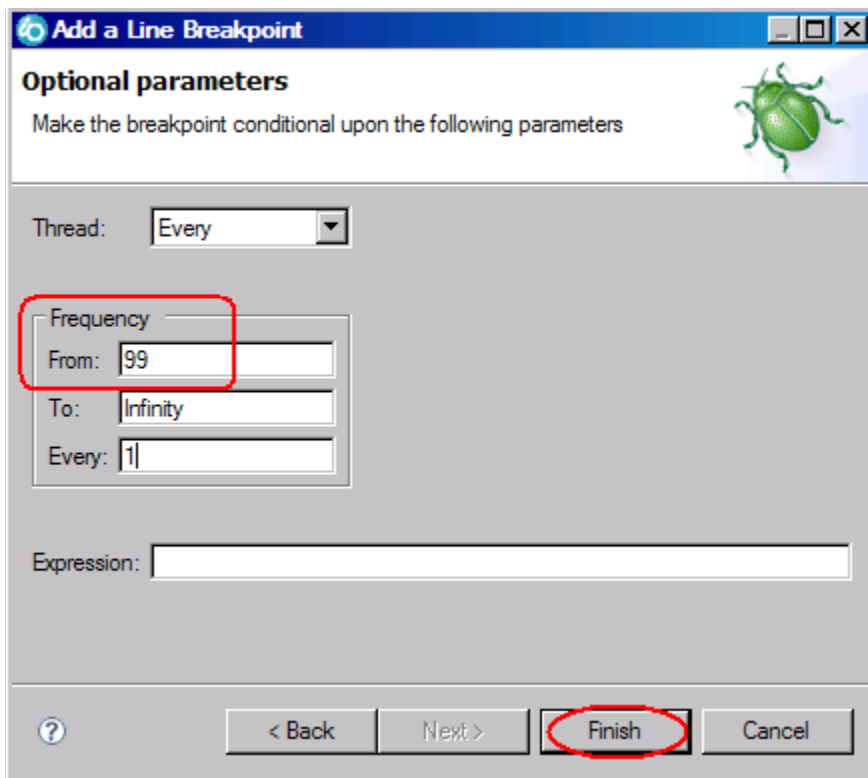
**Tip:**

You can select an existing breakpoint by right-clicking it and selecting **Edit Breakpoint**.

You only want to stop in the loop when it executes for the 99th time or more. You can do that by setting the **From** field of the **Frequency** group to 99.

\_\_6. Under **Frequency** in the **From** field, type 99.

7. Click **Finish**.



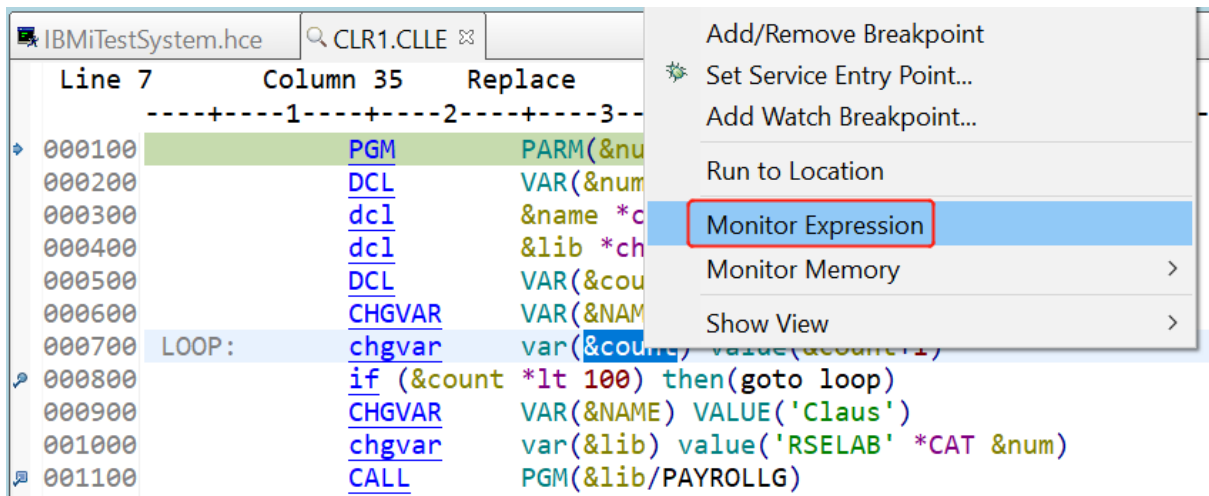
You have added a breakpoint including a conditional breakpoint to your debug session.

## 2.5 Monitoring variables

You can monitor variables in the Monitors view. Now you will monitor the variable `&count`. To monitor a variable:

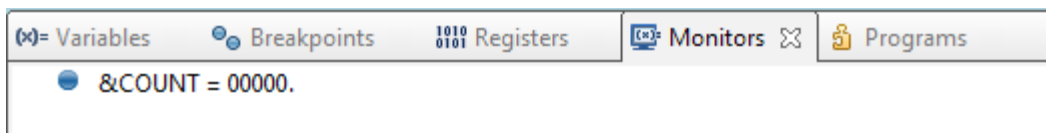
1. In the Source view, double-click the variable `&count`.
2. Right-click `&count`.

3. Click **Monitor Expression** on the pop-up menu.



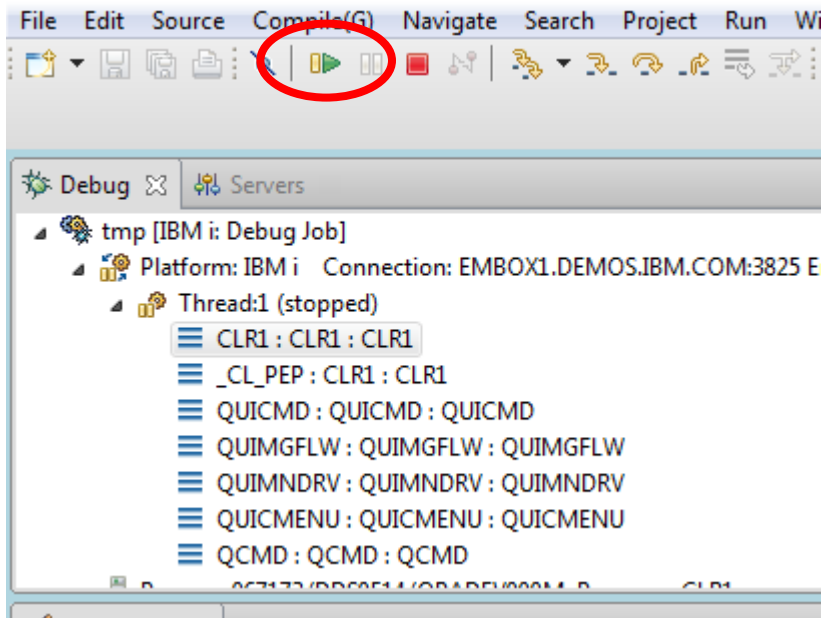
The Monitors view opens.

The variable appears in the Monitors view. Its current value is zero.



Now that some breakpoints and a monitor are set, you can start to run the application.

- \_\_4. Click the **Resume** icon on the Debug toolbar.



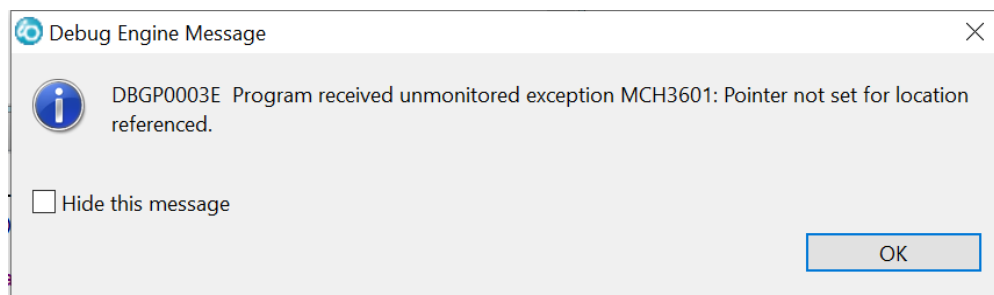
The program starts running and stops at the breakpoint at line 8. Be patient, the Debugger has to stop 98 times but because of the condition continues to run until the 99th time. Notice in the Monitors view, that *&count* now has the value 99.

- \_\_5. Click the **Resume** icon again.  
The program stops at the breakpoint at line 8 again and *&count* now has the value 100.
- \_\_6. Click the **Resume** icon once more so that the program runs to the breakpoint at line 11.
- \_\_7. **If you do not see the error message below, go to “Stepping into a program” topic 2.7.**


## 2.6 Error handling

If you don't have any errors, skip to the next section '**Stepping into a program**'

If you forgot to add the parameter to the CALL program command when calling the program, you will see this error message.

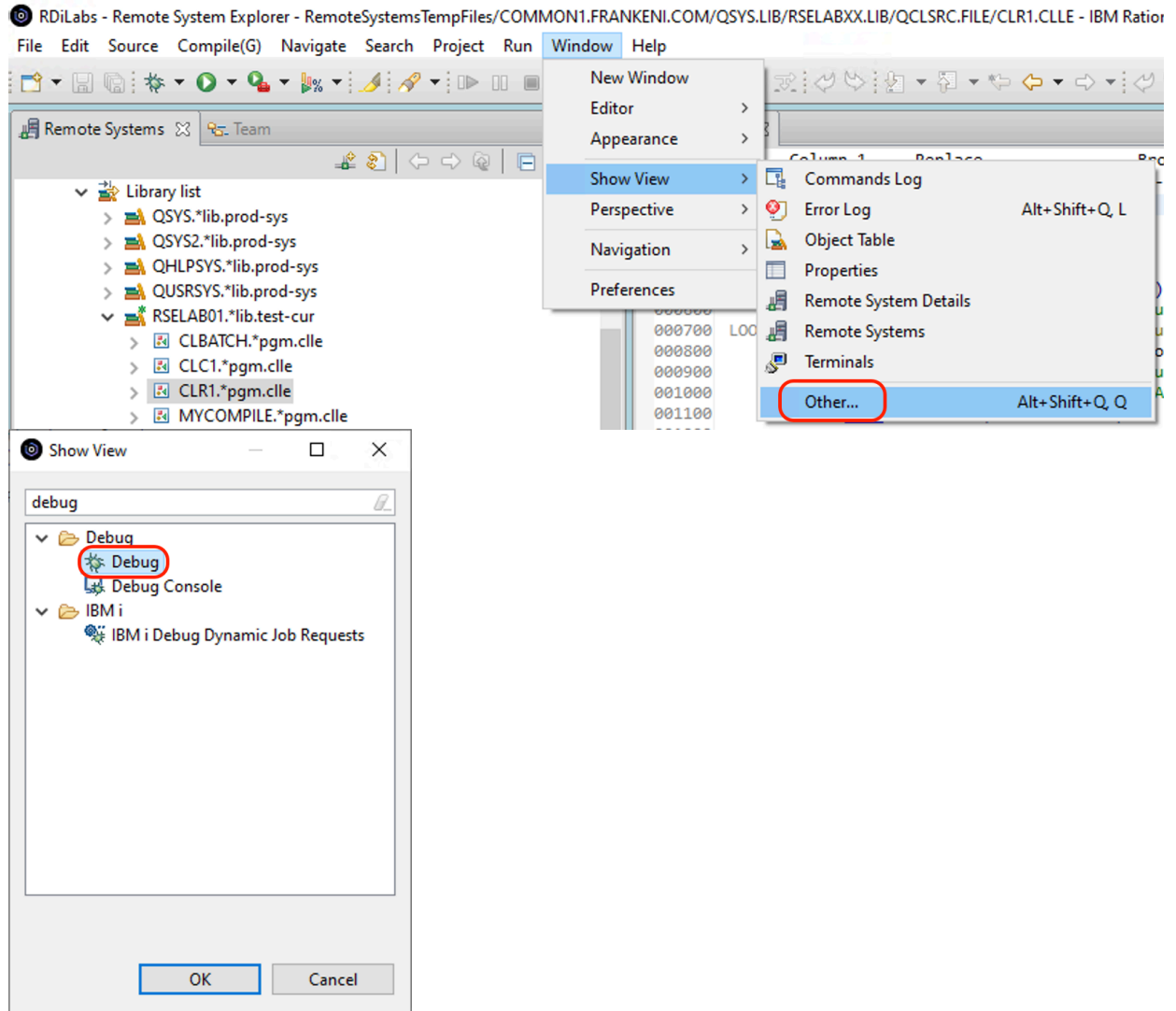


- \_\_1. Click **OK** on the error message dialog.

- \_\_2. Click the **Terminate** icon (  ) on the Debug toolbar  
 The debug session terminates on the workstation but the exception waits for input from the 5250 emulation session.  
 If you closed the Debug view by mistake, you will need to re-open the Debug view and then terminate the debug session on the workstation.

Here are the steps to re-open the debug view:

- \_\_3. Click **Window > Show View > Debug**. Click **Other..** if Debug is not on the list, type Debug in the text box to filter out the options. Select **Debug** under **Debug** will open the view.



Now terminate the Debug session if you haven't done so already.

- \_\_4. Go to your 5250 emulator.  
 \_\_5. Enter C for cancel and press **Enter** until the program messages complete.

```

Display Program Messages

Job 642279/WLAB01/QPADEV025H started on 19/05/14 at 07:52:56 in subsystem QB
Job 642279/WLAB01/QPADEV025H held by user WLAB01 with option SPLFILE(*NO).
Job 642279/WLAB01/QPADEV025H released by user WLAB01.
Job 642279/WLAB01/QPADEV025H held by user WLAB01 with option SPLFILE(*NO).
Job 642279/WLAB01/QPADEV025H released by user WLAB01.
MCH3601 received by procedure CLR1. (C D I R)
C
Job 642279/WLAB01/QPADEV025H held by user WLAB01 with option SPLFILE(*NO).
Job 642279/WLAB01/QPADEV025H released by user WLAB01.
MCH3601 received by procedure CLR1. (C D I R)

Type reply, press Enter.
Reply . . .

F3=Exit  F12=Cancel

MA a MW 20/018

```

In the workbench:

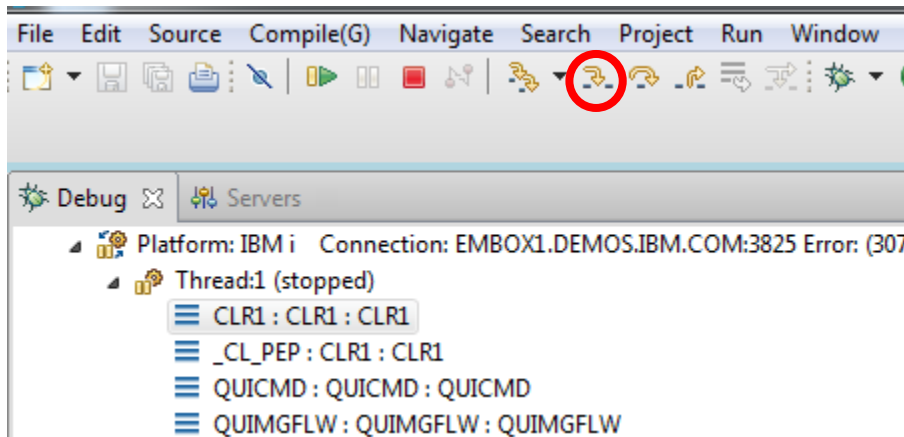
- \_\_6. Click the **Remove all terminated launches** icon on the Debug toolbar to clean up the Debug view.  
To restart the program and start the debug session again.  
On the 5250 command line, call the CLR1 program with the parameter xx.
- \_\_7. Enter: **CALL PGM(RSELABxx/CLR1) PARM('xx')** where xx is your team number.

## 2.7 Stepping into a program

The Debugger allows you to step over a program call or step into it. When you step over a program call, the called program runs and the Debugger stops at the next executable statement in the calling program. You are going to step into the PAYROLLG program.

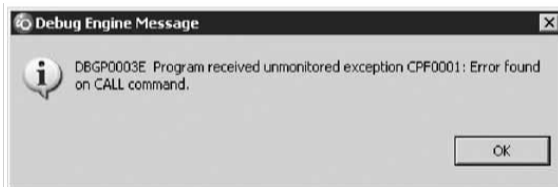
To step into a program:

1. Click the **Step into** icon on the Debug toolbar.



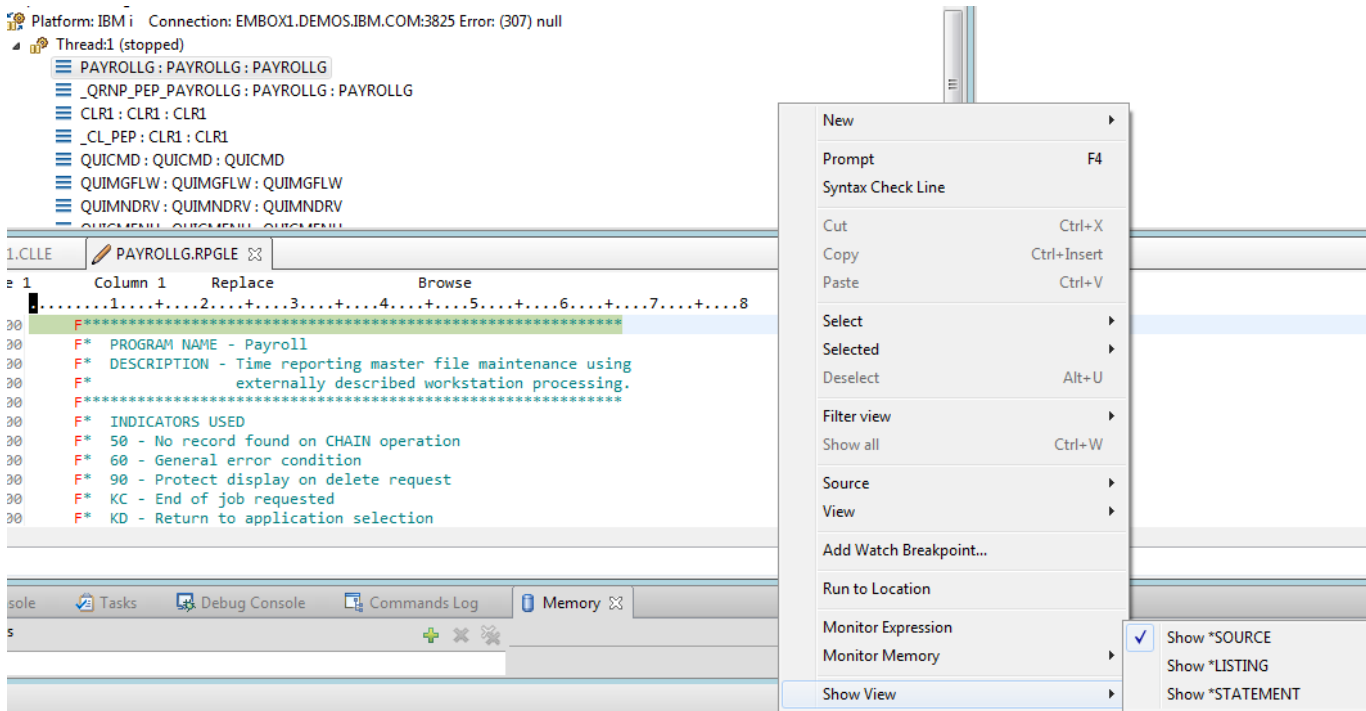
The source of PAYROLLG is displayed. Depending on the option you used to compile the program (\*SRCDBG or \*LSTDBG for RPG, or \*SOURCE, \*LIST, or \*ALL for ILE RPG), this window displays either the Source or Listing View.

If you specified an incorrect parameter for the CALL program command, or your library list does not include RSELABxx, you will see this error message.



Make sure your library list is correct and complete the same steps as covered in the section called **Error Handling** in topic 2.6.

2. Right-click anywhere in the Source view and click **Show view > Show \*LISTING** on the pop-up menu.



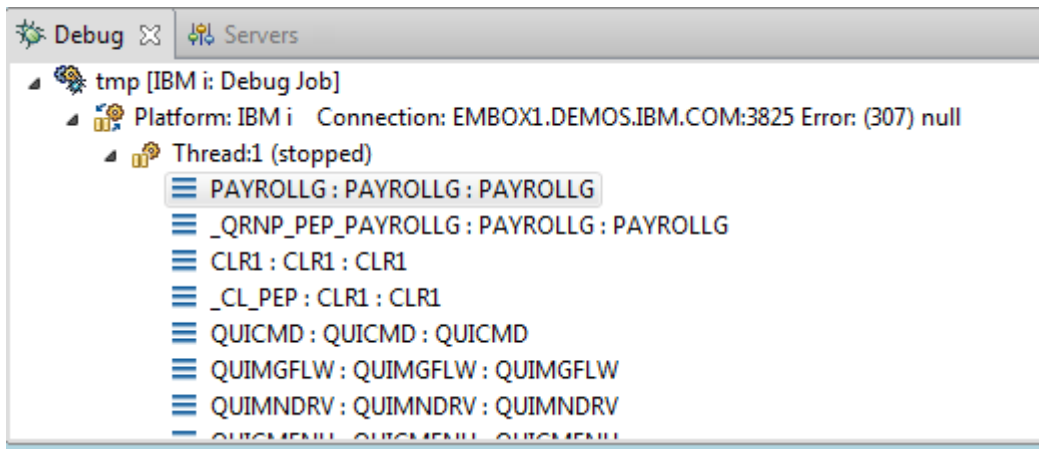
3. Page down in the source and take a look at the expanded file descriptions. You don't have any /Copy member in your PAYROLLG program, but these would also be shown in a Listing view. Switch back to the Source view.
4. Right-click anywhere in the Source view.
5. Click **Switch view > Show \*SOURCE** on the pop-up menu. You have stepped into PAYROLLG program, switched the view from source to listing and back to source.

## 2.8 Listing call stack entries

The Debug view in the upper left pane, lists all call stack entries. It contains a tree view for each thread. The thread can be expanded to show every program, module, and procedure that is on the stack at the current execution point. If you double-click on a stack entry you will display the corresponding source if it is available. Otherwise, the message No Debug data available appears in the Source view.



- \_\_1. In the Debug view, expand the stack entry of Thread1 if it is not expanded already.



The stack entry allows you to work with and switch between different programs and/or ILE modules.

You have viewed the call stack entries of your program.

## 2.9 Setting breakpoints in PAYROLLG

Now you add some breakpoints in PAYROLLG.

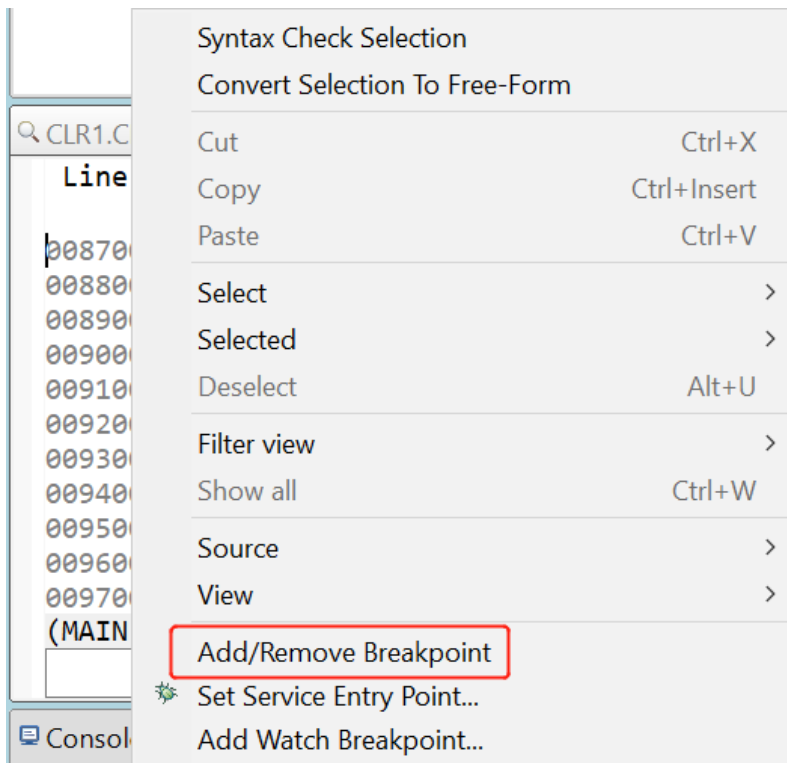
To add breakpoints:

- \_\_1. Select PAYROLLG in Thread1.
- \_\_2. In the source view scroll to line 57.
- \_\_3. Double-click the prefix area (to the left of the source code line number) of line 57.  
A breakpoint icon is added to the prefix area of this line to indicate that a breakpoint is set.

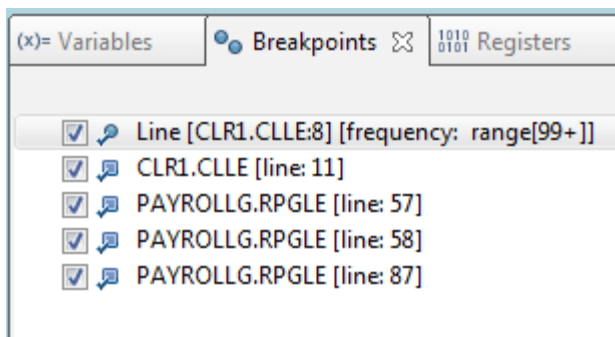
4. Repeat the above step for line 58.

Line 57	Column 1	Replace	Browse
		.....CL0N01Factor1+++++Opcode(E)+Extended-factor2+++++.....	
005200	C	MAIN	BEGSR
005300	C		dou *INKC
005400	C		EVAL *IN60 = *OFF
005500	C		EVAL EMESS = *BLANK
005600	C		EVAL EMPAPL = *BLANK
005700	C		EVAL PRJAPL = *BLANK
005800	C		EVAL RSNAPL = *BLANK
005900	C*		
006000	C*	Write the SELECT format to display. If end of job requested,	
006100	C*		
006200	C*		

- \_\_5. Right-click in the prefix area of line 87 and click **Add/Remove Breakpoint** on the pop-up menu.



To view all breakpoints, select the **Breakpoints** tab from the top left pane:



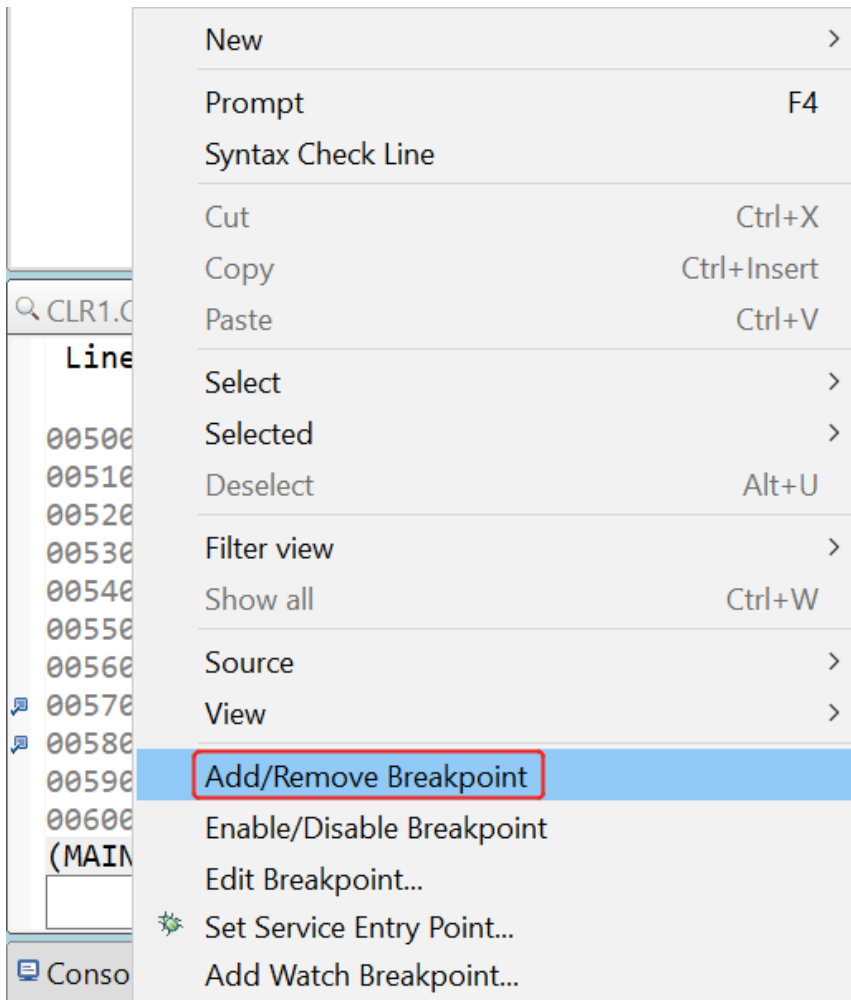
This view shows all breakpoints currently set in your Debug session. This is a convenient place to work with breakpoints. You can remove, disable/enable, add, or edit a breakpoint. These tasks are available from the pop-up menu when you right-click in the view area. Double-click any entry to show the source where the breakpoint is set.

## 2.10 Removing a breakpoint in PAYROLLG

It is also easy to remove breakpoints.  
To remove a breakpoint:

- \_\_1. Right-click the prefix area of line 58.

2. Click **Add/Remove Breakpoint** on the pop-up menu.



The icon is removed from the prefix area indicating that no breakpoint is set on that line. The breakpoint is also removed from the list in the Breakpoints view.

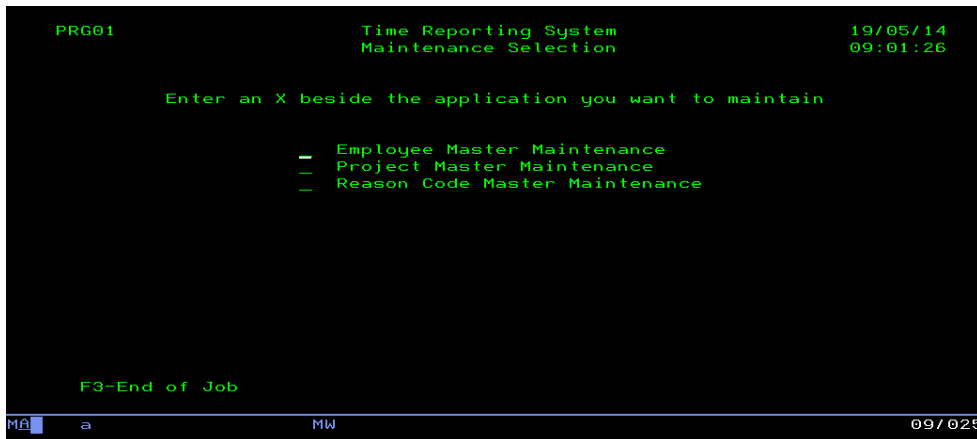


**Tip:**

Double-clicking on a breakpoint in the prefix area will also remove that breakpoint.

3. Now you are ready to run the PAYROLLG program.  
Click the **Resume** icon (▶) from the Debug toolbar.  
The program starts running and runs to the breakpoint at line 57.

- \_\_4. Click the **Resume** icon again.  
The program waits for input from the 5250-emulation session.



- \_\_5. Type an X beside the **Project Master Maintenance** option.  
\_\_6. Press **Enter** in the emulation session. The program runs to the breakpoint at line 87.  
You have removed a breakpoint from PAYROLLG and started to run the program.

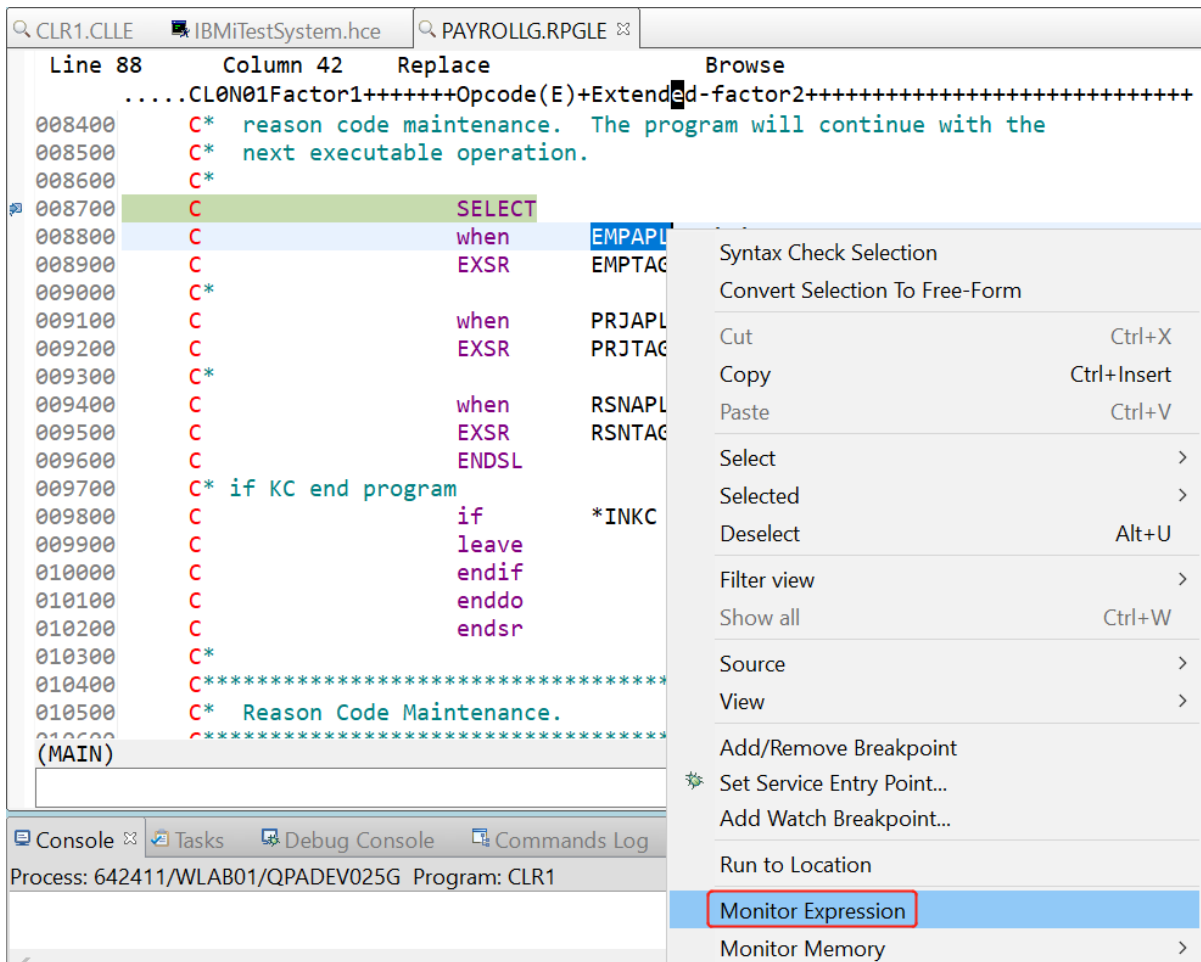
## 2.11 Monitoring variables in PAYROLLG

Now let's monitor variables and change them in PAYROLLG.

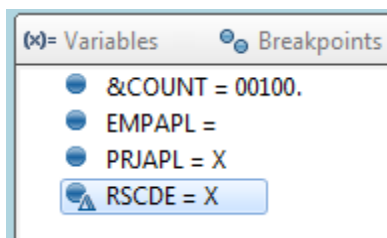
To monitor variables:

- \_\_1. In the source view, double-click the variable EMPAPL on line 88.  
\_\_2. Right-click the variable.

3. Click **Monitor Expression** on the pop-up menu.



- 4. Click the **Monitors** tab in the upper right pane. The variable appears in the **Monitors** view. Its value is blank because you did not select the **Employee Master Maintenance** option.
- 5. In the same way add the variables PRJAPL on line 91 and RSCDE on line 113 to the monitor. Variable PRJAPL equals X because you did select the **Project Master Maintenance** option.
- 6. In the Monitors view, double-click the variable RSCDE. The value changes into an entry field.
- 7. In the entry field, type in the new value X for the variable.



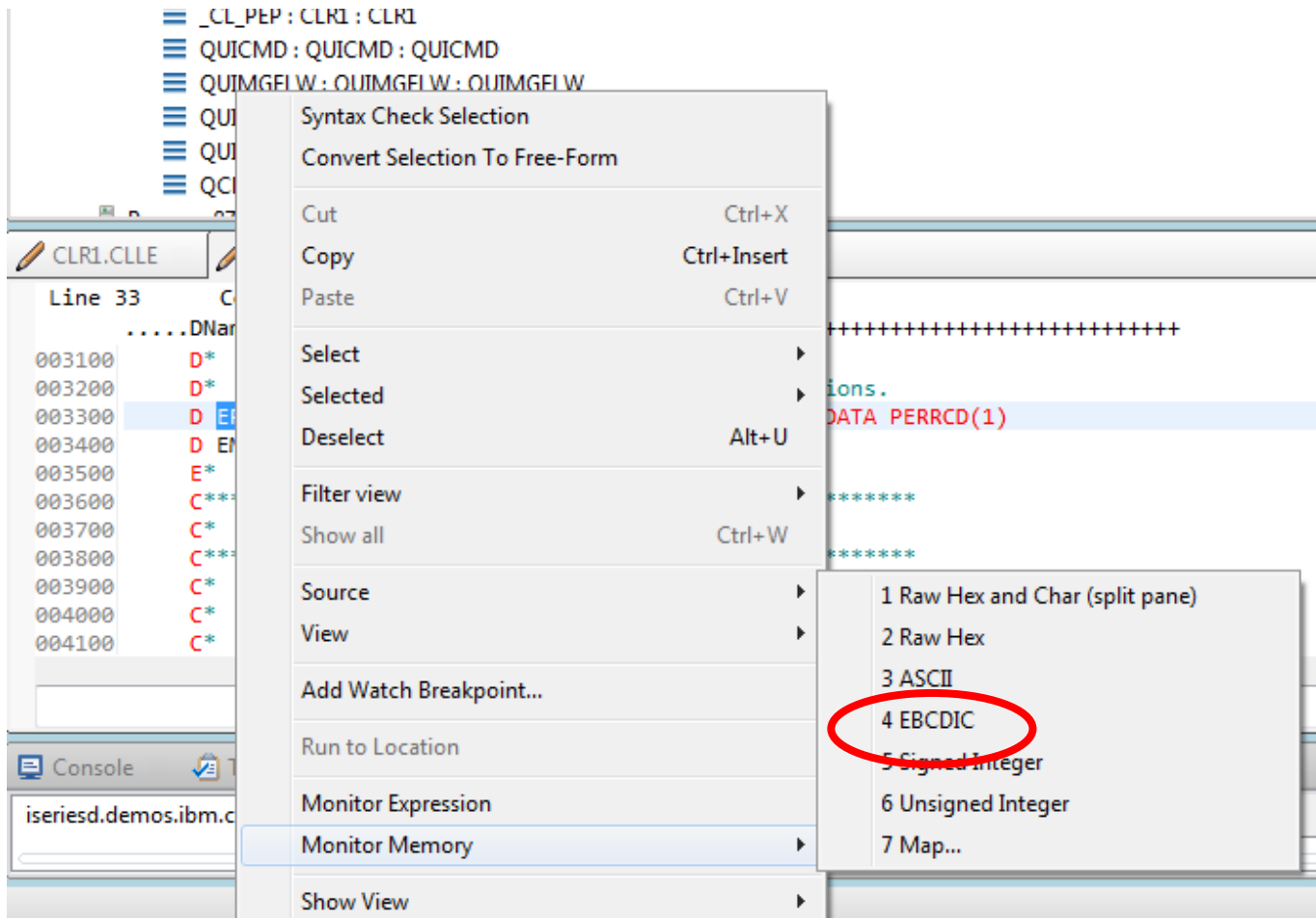
- 8. Press **Enter**.  
The variable is successfully changed.

## 2.12 Adding a memory monitor

Adding a memory monitor for a variable allows you to view the memory starting with the address where the variable is located. The memory can be displayed in different formats, for example hexadecimal and text.

To add a memory monitor:

1. In the Source view, double-click the variable `ERR` in line 33.
2. Right-click and select **Monitor Memory > EBCDIC** on the pop-up menu.



This will open the Memory view in the pane at the bottom of the perspective. The tab shows the name of the variable.

Address	0 - 3	4 - 7	8 - B	C - F
EE65E6468100FB60	?...	?...	?...	?...
EE65E6468100FB70	?...	?...	?...	?...
EE65E6468100FB80	M	AIN	ENAN	CE S
EE65E6468100FB90	ELEC	TION	COD	E NO
EE65E6468100FBA0	T EQ	UAL	TO "	X"
EE65E6468100FBB0	MO	RE T	HAN	ONE
EE65E6468100FBC0	APPL	ICAT	ION	SELE
EE65E6468100FBD0	CTED	FOR	MAI	NTEN
EE65E6468100FBE0	ANCE		NO	APPL
EE65E6468100FBF0	ICAT	ION	SELE	CTED
EE65E6468100FC00	FOR	MAI	NTEN	ANCE

- \_\_3. Use the scroll bar on the right of the Memory view to scroll down. You can see the current content of the memory.
- \_\_4. Right-click in the view area.
- \_\_5. Click **Reset to Base Address** on the pop-up menu to return to the starting address.
- \_\_6. To get the hex content of the memory starting with the selected variable, click the tab **New Renderings** and select **Raw Hex** for example. A new page with the hex values is added to the Memory view.



- \_\_7. Click the Toggle Split Pane icon to display the character values as well.

Address	0 - 3	4 - 7	8 - B	C - F
EE65E6468100FB80	404040D4	C1C9D5E3	C5D5C1D5	C3C540E2
EE65E6468100FB90	C5D3C5C3	E3C9D6D5	40C3D6C4	C540D5D6
EE65E6468100FBA0	E340C5D8	E4C1D340	E3D6407F	E77F4040
EE65E6468100FBB0	4040D4D6	D9C540E3	C8C1D540	D6D5C540
EE65E6468100FBC0	C1D7D7D3	C9C3C1E3	C9D6D540	E2C5D3C5
EE65E6468100FBD0	C3E3C5C4	40C6D6D9	40D4C1C9	D5E3C5D5
EE65E6468100FBE0	C1D5C3C5	40404040	40D5D640	C1D7D7D3
EE65E6468100FBF0	C9C3C1E3	C9D6D540	E2C5D3C5	C3E3C5C4
EE65E6468100FC00	40C6D6D9	40D4C1C9	D5E3C5D5	C1D5C3C5
EE65E6468100FC10	40404040	40404040	4040C1C3	E3C9D6D5
EE65E6468100FC20	40C3D6C4	C540D5D6	E340C5D8	E4C1D340
EE65E6468100FC30	E3D6407F	C17F6B40	7FC37F40	D6D9407F
EE65E6468100FC40	C47F4040	40404040	40C1C4C4	40D9C5D8
EE65E6468100FC50	E4C5E2E3	C5C440C2	E4E340D9	C5C3D6D9
FF65E6468100FC60	C440C1D3	D9C5C1C4	F840C5F7	C9E2F3E2

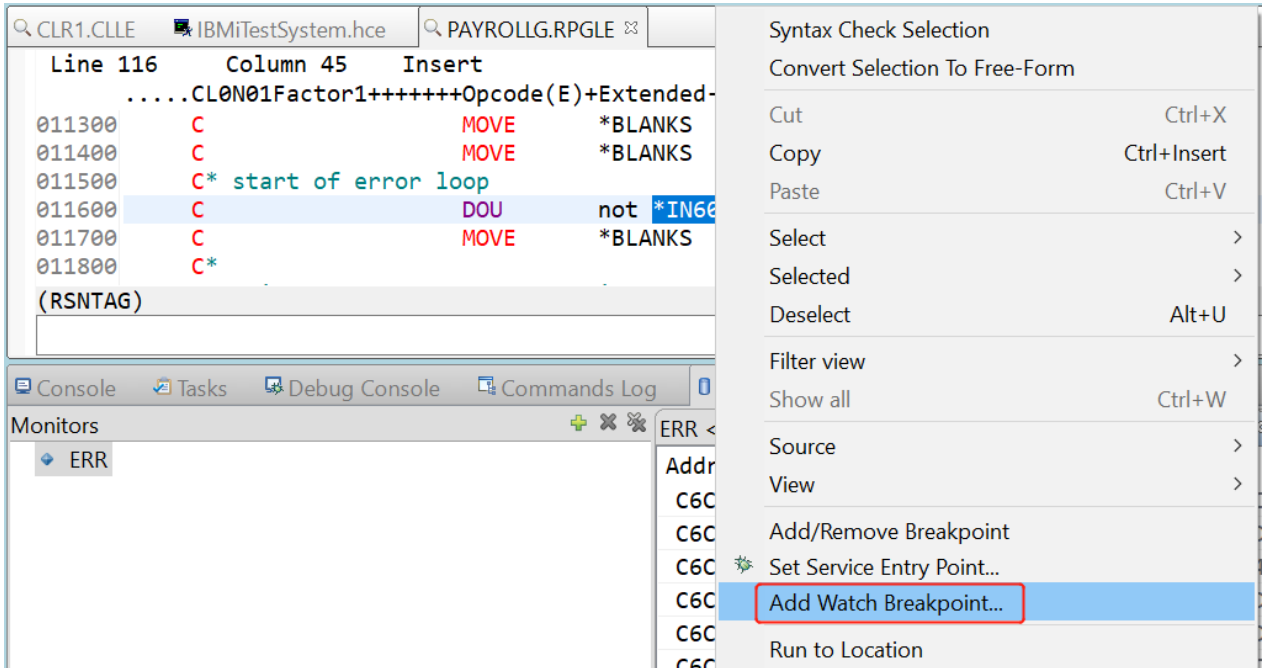
You have added a memory monitor for the variable ERR.

## 2.13 Setting Watch breakpoints

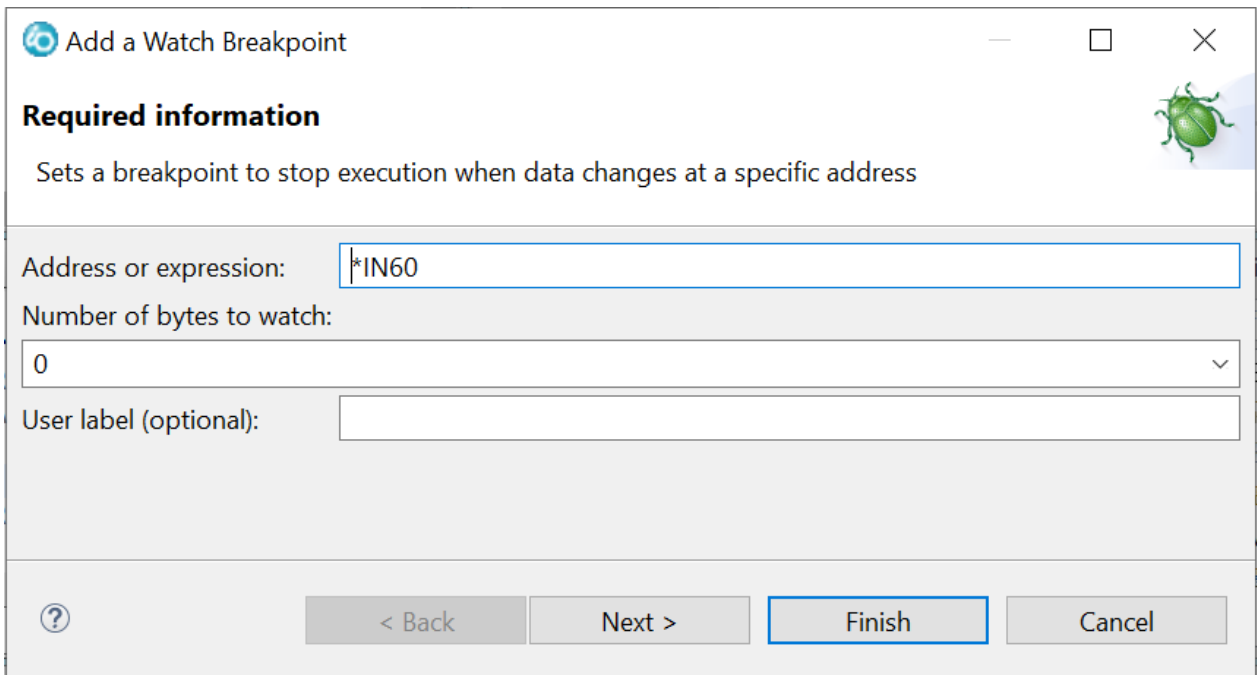
A Watch breakpoint provides a notification to the user when a variable changes. It will suspend the execution of the program until an action is taken.

To set a Watch breakpoint:

- \_\_1. Go to the **Line number** field at the bottom of the source area. In this field enter 116 to go to that line.
- \_\_2. Double-click variable **\*IN60** to highlight it.
- \_\_3. Right-click and click **Add Watch Breakpoint** on the pop-up menu.

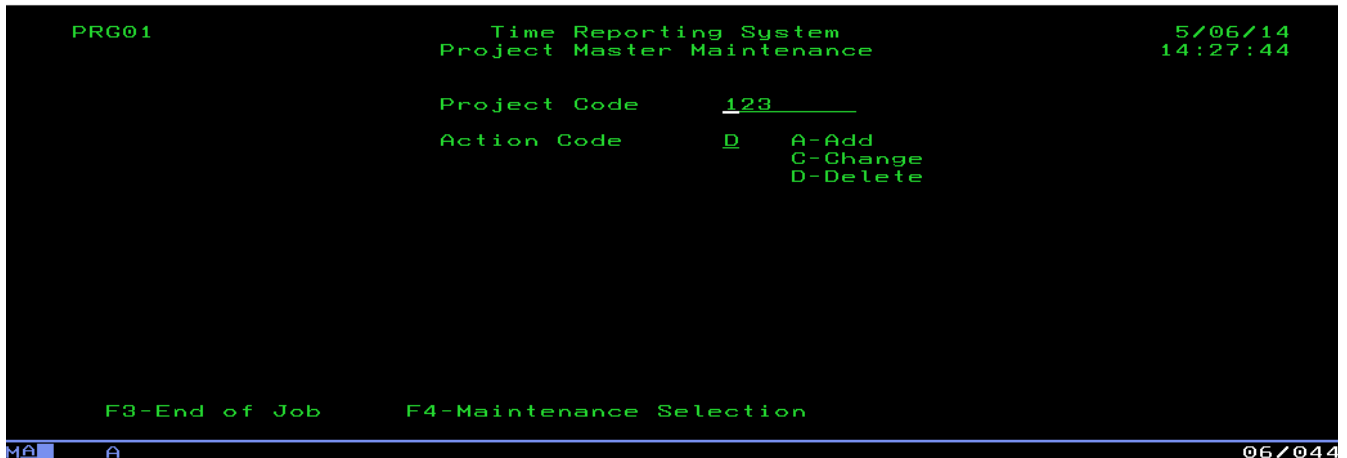


The Add a Watch Breakpoint window opens. The **Expression** field is pre-filled with the highlighted variable **\*IN60**. By default, the Number of bytes to watch field is set to zero, which means the variable will be watched in its defined length.



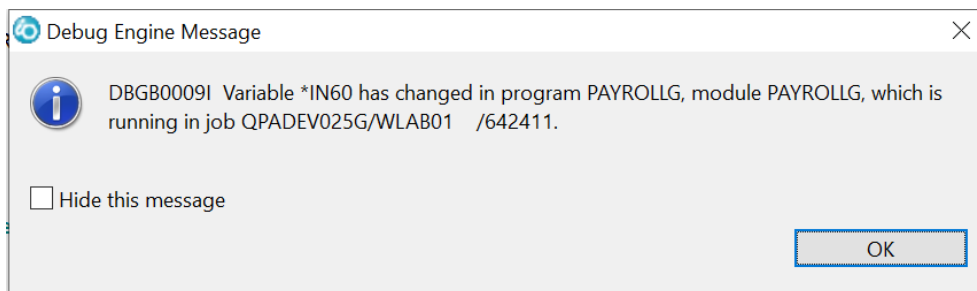
4. Click **Finish**. The Watch breakpoint is now set.

- \_\_5. Click the **Resume** button on the Debug toolbar.  
The application waits for input from the 5250-emulation session.



In the 5250 emulation session, type:

- \_\_6. 123 for **Project Code** and D (for delete) in the **Action Code** field.  
\_\_7. Press **Enter**.  
A message is displayed indicating that the variable **\*IN60** has changed.



- \_\_8. Click **OK**. The program stops at line **465**. This line is located immediately after the statement which caused the variable **\*IN60** to change.

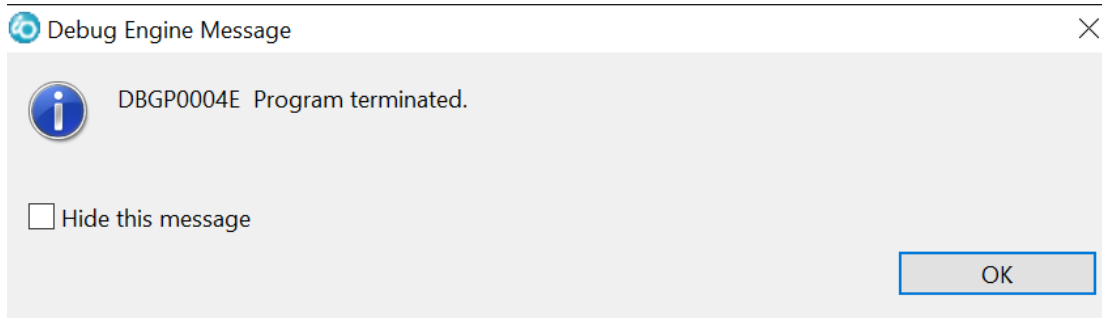
You have added a Watch breakpoint for the variable **\*IN60** and run the program to see the notification that the variable has changed.

## 2.14 Terminate a debug session

To close the debugger:

- \_\_1. Click the **Resume** icon on the Debug toolbar. The application waits for input from the 5250 emulation session.  
\_\_2. Switch to the 5250 emulation session.

- \_\_3. Press **F3** to end the program.  
A message **Program terminated** appears:



- \_\_4. Click **OK**.

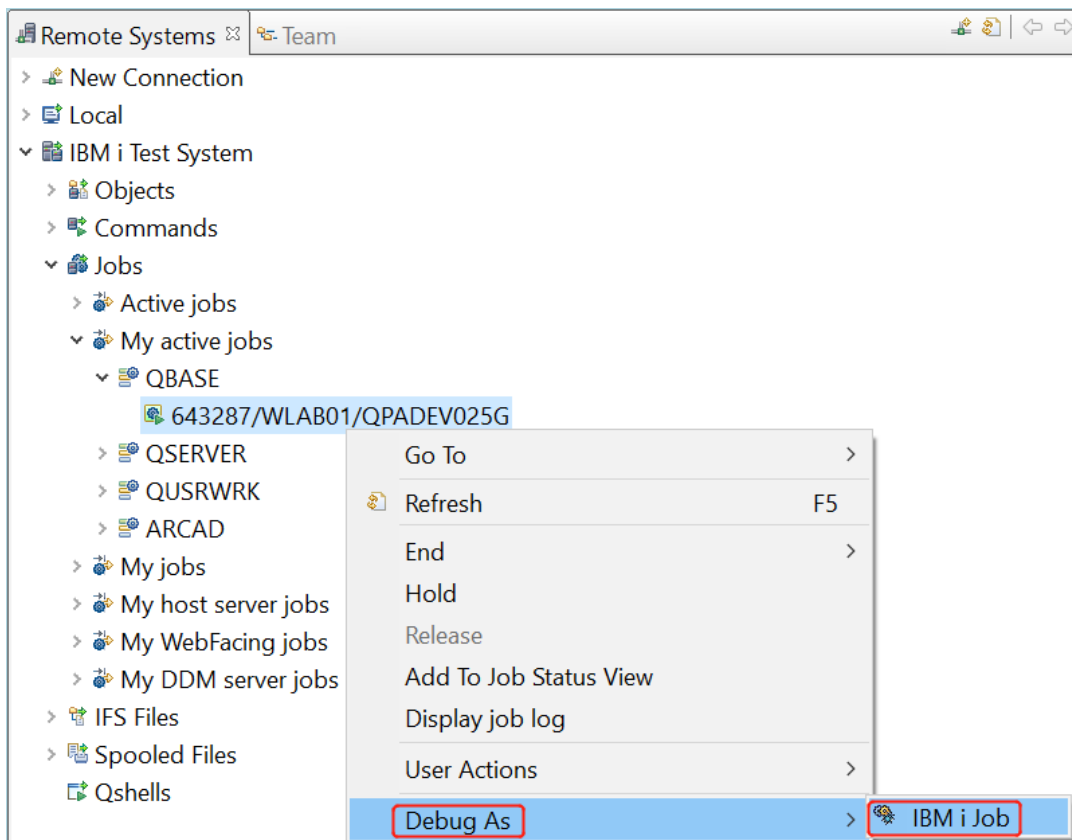
### 3 Debugging a Job

In addition to being able to debug a program, you can also debug a job.

To debug a job:

In the Remote Systems Explorer perspective, under your active server connection, **IBM i Test System:**

- \_\_1. Expand **Jobs > My Active Jobs > QBASE**.  
If you cannot find the job in **QBASE**, then try to expand **Jobs > My Active Jobs > QINTER**.
- \_\_2. Right-click the active job under **QBASE (or QINTER)** and select **Debug As > IBM i Job**.

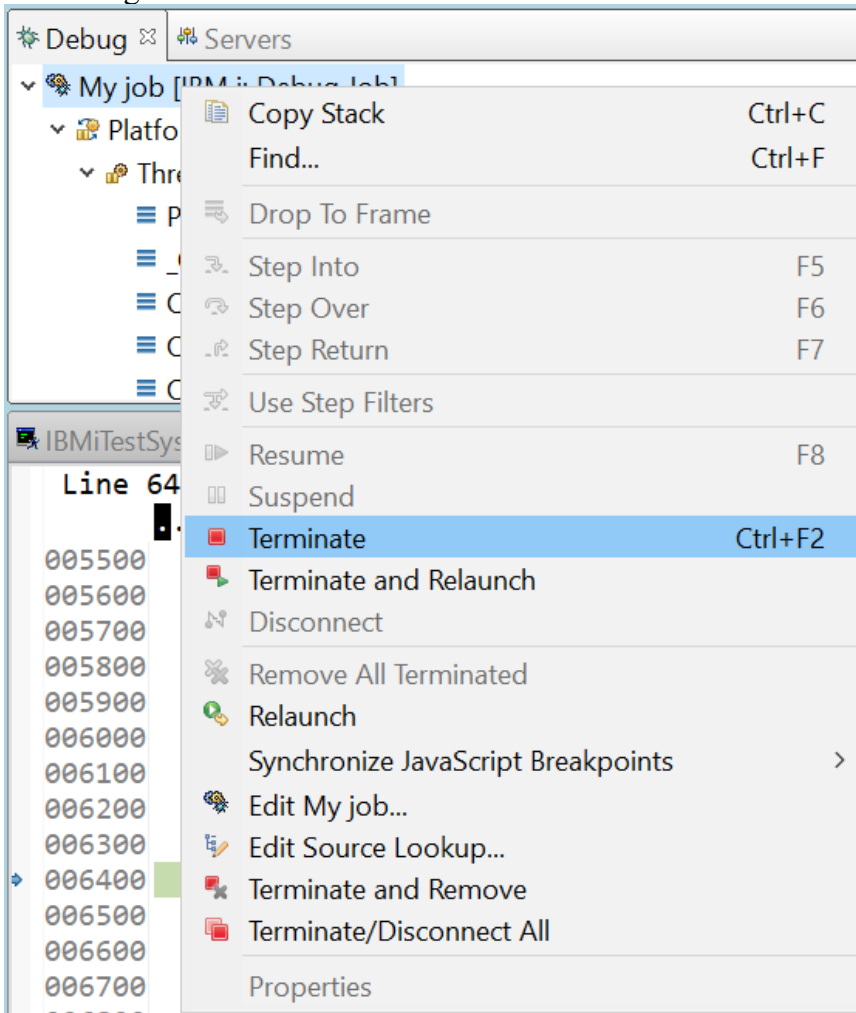


The debug session begins and connects you to the running application.

You can set breakpoints, monitor variables and memory in the same way you did before.

3. Terminate the debug session by right-clicking the job in the Debug view and selecting **Terminate** from the pop-up menu.

The debug session is terminated.



## 4 Lab summary

In this module, you learned how to debug a program using the Integrated IBM i Debugger.

### Lessons learned

- Start a debug session using service entry points
- Add a breakpoint
- Add a conditional breakpoint
- Edit a breakpoint
- Monitor a variable in the Monitors view
- Step into your payroll program
- Show a Listing view
- Display source from call stack entries
- View all breakpoints
- Remove a breakpoint
- Monitor memory
- Set a Watch breakpoint
- Close the debugger
- Invoke the debugger from a Debug Configurations window.

This tutorial has taught you so far how to maintain a payroll application using the Remote Systems Explorer. You learned how to start the product and open the Remote Systems Explorer perspective and how to use tools and views in this perspective to connect to an IBM i system and edit, verify, compile and debug the payroll application.

## Congratulations!

You have successfully completed the Debug lab exercises.

We recommend that you move on to the next lab in the sequence; or browse the list of labs on Rational Developer for i - Hands-On Labs at [http://ibm.biz/rdi\\_labs](http://ibm.biz/rdi_labs) to choose a lab of interest.

More information, material and opportunities to discuss the product can be found at our RDi Hub:

[http://ibm.biz/rdi\\_hub](http://ibm.biz/rdi_hub)



---

## Appendix A Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

---

## Appendix B Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
System z	Tivoli	WebSphere	Workplace	System p	

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product and service names may be trademarks or service marks of others.



---

© Copyright IBM Corporation 2021

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [ibm.com/legal/copytrade.shtml](http://ibm.com/legal/copytrade.shtml)

Other company, product and service names may be trademarks or service marks of others.



Please Recycle