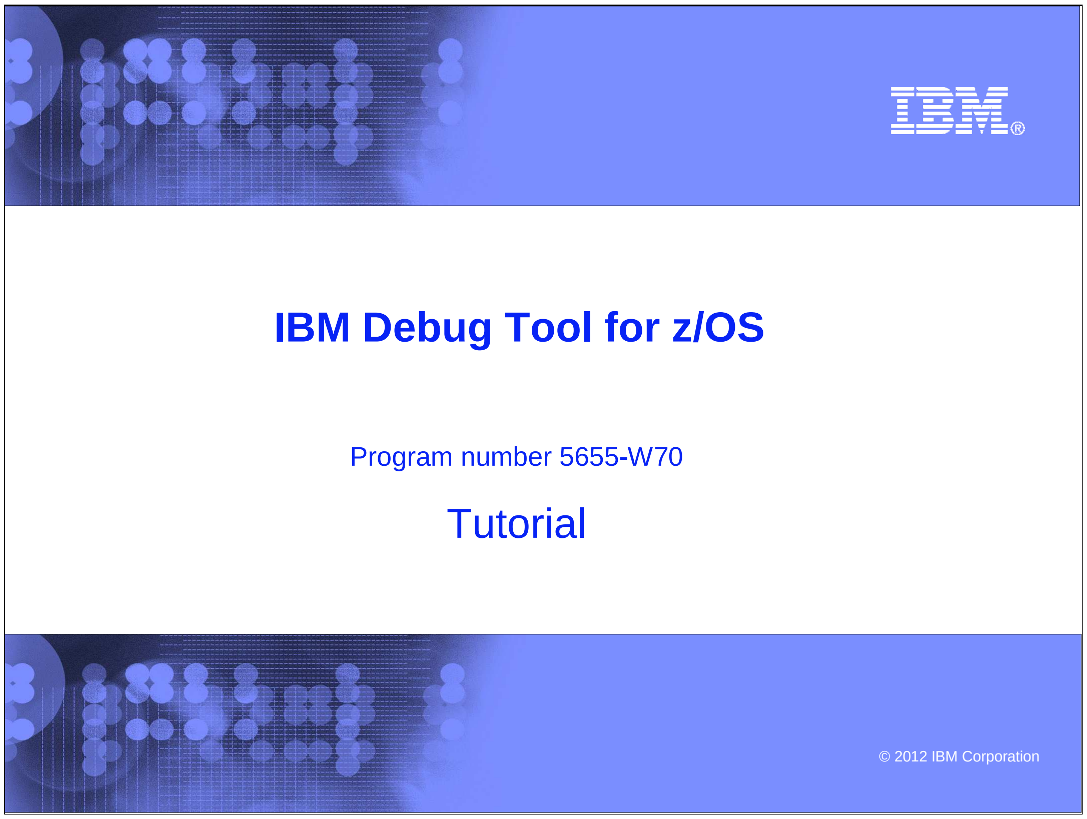
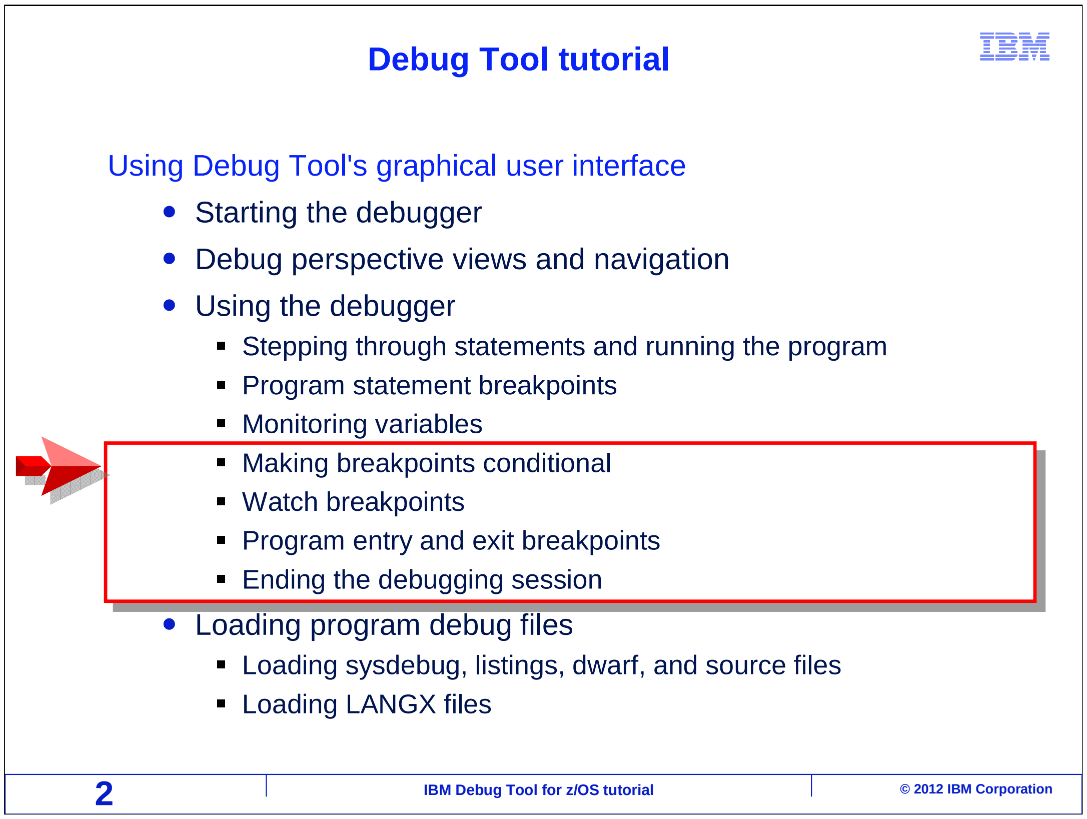


IBM Debug Tool for z/OS

Program number 5655-W70

Tutorial

This is the tutorial for IBM Debug Tool for z/OS®, one of the IBM zSeries® problem determination tools.



Debug Tool tutorial

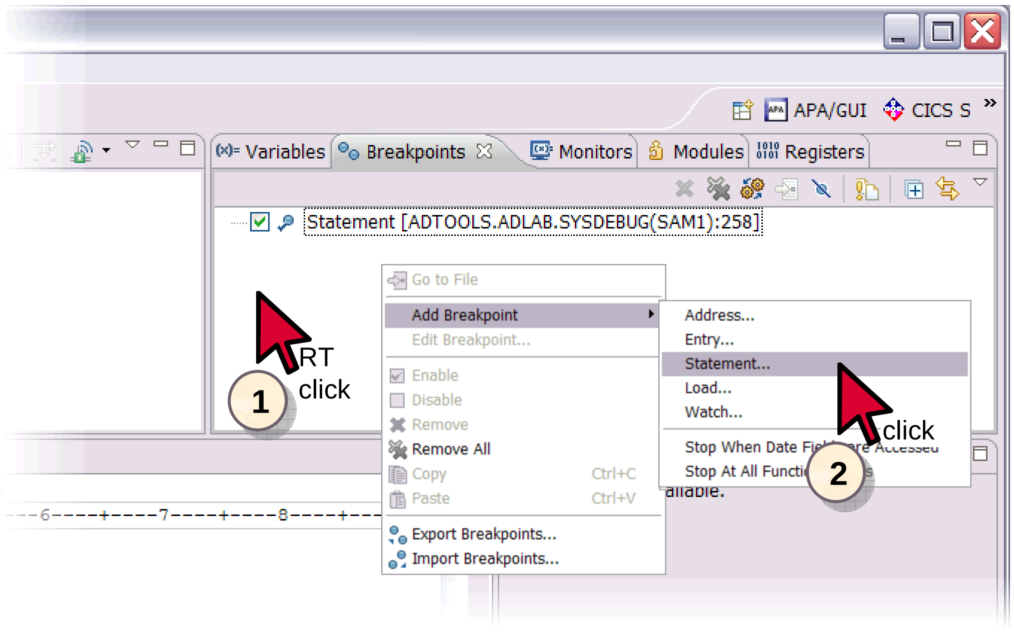
Using Debug Tool's graphical user interface

- Starting the debugger
- Debug perspective views and navigation
- Using the debugger
  - Stepping through statements and running the program
  - Program statement breakpoints
  - Monitoring variables
  - Making breakpoints conditional
  - Watch breakpoints
  - Program entry and exit breakpoints
  - Ending the debugging session
- Loading program debug files
  - Loading sysdebug, listings, dwarf, and source files
  - Loading LANGX files

2 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

In this section you will see how breakpoints can be made conditional, how to set watch breakpoints, how to set breakpoints that will stop when a program or sub-program is entered, and how to end a debugging session.

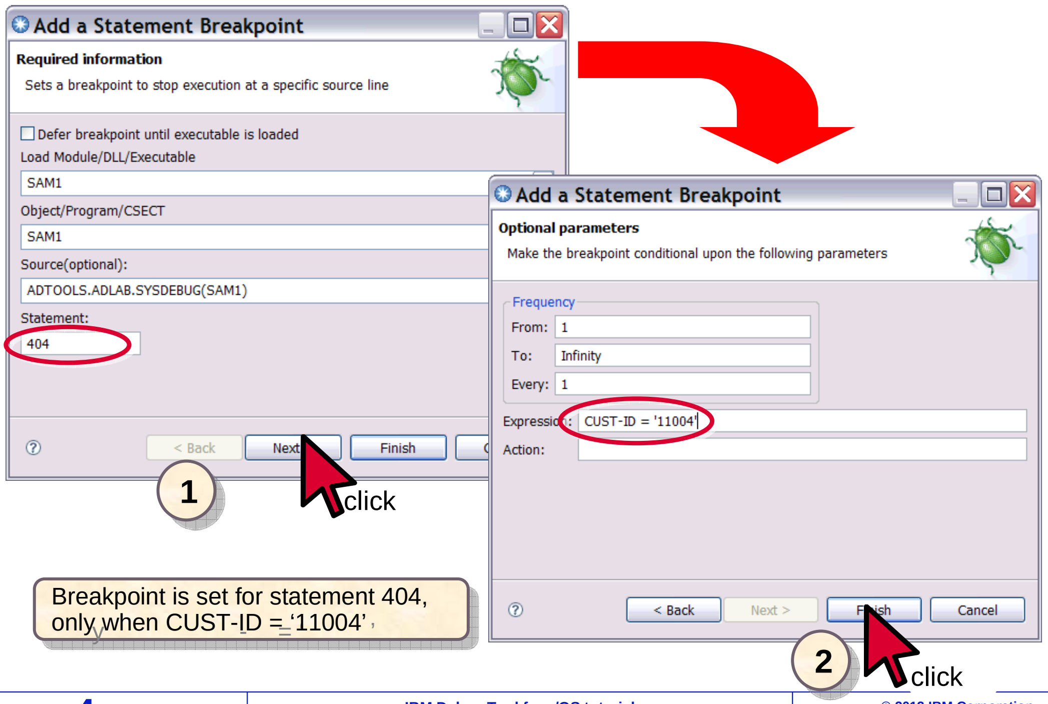
Begin by creating a breakpoint



Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]

To define a conditional breakpoint, right-click in the white area of the breakpoints view. Then select "add breakpoint", and select the type of breakpoint that you want to add. In this example a statement breakpoint is created.

Specify an expression to set a conditional breakpoint


Add a Statement Breakpoint

Required information

Sets a breakpoint to stop execution at a specific source line

☐ Defer breakpoint until executable is loaded

Load Module/DLL/Executable: SAM1

Object/Program/CSECT: SAM1

Source(optional): ADTOOLS.ADLAB.SYSDEBUG(SAM1)

Statement: 404

Add a Statement Breakpoint

Optional parameters

Make the breakpoint conditional upon the following parameters

Frequency

From: 1

To: Infinity

Every: 1

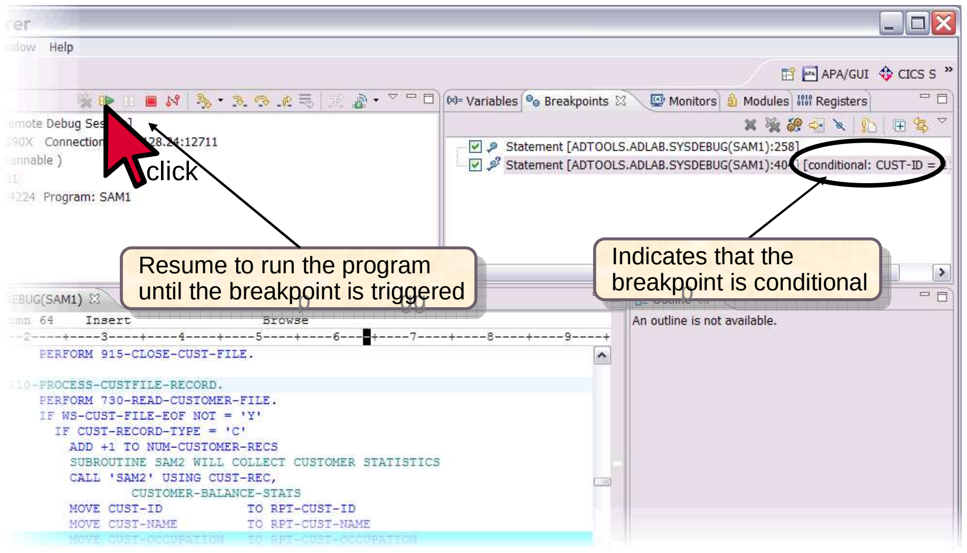
Expression: CUST-ID = '11004'

Action:

Breakpoint is set for statement 404, only when CUST-ID = '11004'

Specify the statement where the breakpoint will be set, in this case "404", and click "next". In the second screen add a conditional expression. This expression specifies that CUST-ID must equal a specific value before the breakpoint can trigger. If CUST-ID does not match the value when statement 404 runs, the breakpoint will not trigger. Click Finish.

Run the program


Help
APA/GUI
CICS S
Variables
Breakpoints
Monitors
Modules
Registers
click
Resume to run the program until the breakpoint is triggered
Indicates that the breakpoint is conditional

```
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):40 [conditional: CUST-ID =
```

```
emote Debug Ses
90X  Connection  28.24:12711
unnable )
224  Program: SAM1
```

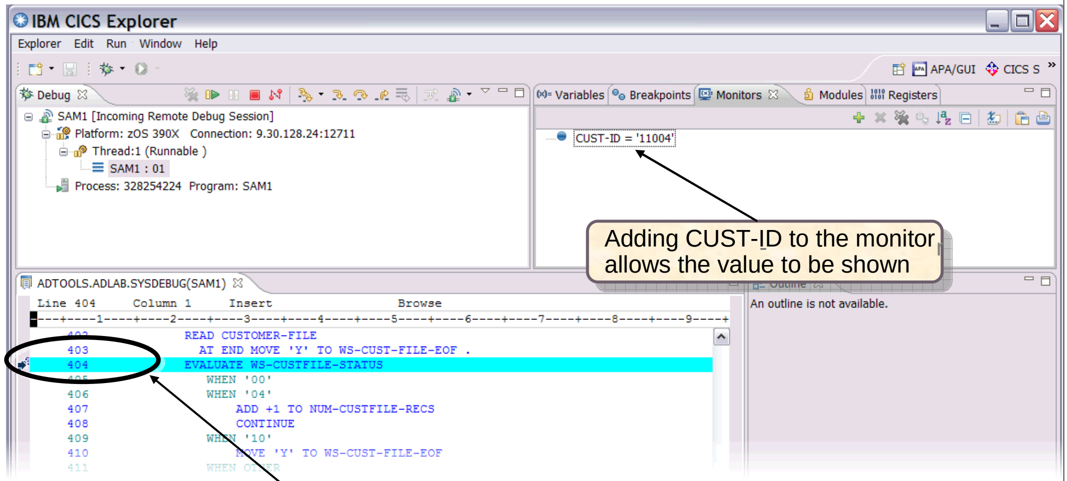
An outline is not available.

```
EBUG(SAM1)
umn 64     Insert                       Browse
--2----+----3----+----4----+----5----+----6---█+----7----+----8----+----9----+
    PERFORM 915-CLOSE-CUST-FILE.

10-PROCESS-CUSTFILE-RECORD.
    PERFORM 730-READ-CUSTOMER-FILE.
    IF WS-CUST-FILE-EOF NOT = 'Y'
      IF CUST-RECORD-TYPE = 'C'
        ADD +1 TO NUM-CUSTOMER-RECS
        SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
        CALL 'SAM2' USING CUST-REC,
                CUSTOMER-BALANCE-STATS
        MOVE CUST-ID           TO RPT-CUST-ID
        MOVE CUST-NAME         TO RPT-CUST-NAME
        MOVE CUST-OCCUPATION   TO RPT-CUST-OCCUPATION
```

The breakpoint was added, and appears in the breakpoints view, and is shown to be conditional. Click “resume” to run the program.

The program stopped the next time it reached statement 404 and the condition was true


IBM CICS Explorer
Explorer Edit Run Window Help
APA/GUI CICS S
Debug
Variables Breakpoints Monitors Modules Registers
Outline

SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:12711
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

CUST-ID = '11004'

ADTOOLS.ADLAB.SYSDEBUG(SAM1)

```
Line 404     Column 1     Insert                    Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+
402               READ CUSTOMER-FILE
403                 AT END MOVE 'Y' TO WS-CUST-FILE-EOF .
404               EVALUATE WS-CUSTFILE-STATUS
405                  WHEN '00'
406                  WHEN '04'
407                      ADD +1 TO NUM-CUSTFILE-RECS
408                      CONTINUE
409                  WHEN '10'
410                      MOVE 'Y' TO WS-CUST-FILE-EOF
411                  WHEN OTHER
```

An outline is not available.

Adding CUST-ID to the monitor allows the value to be shown

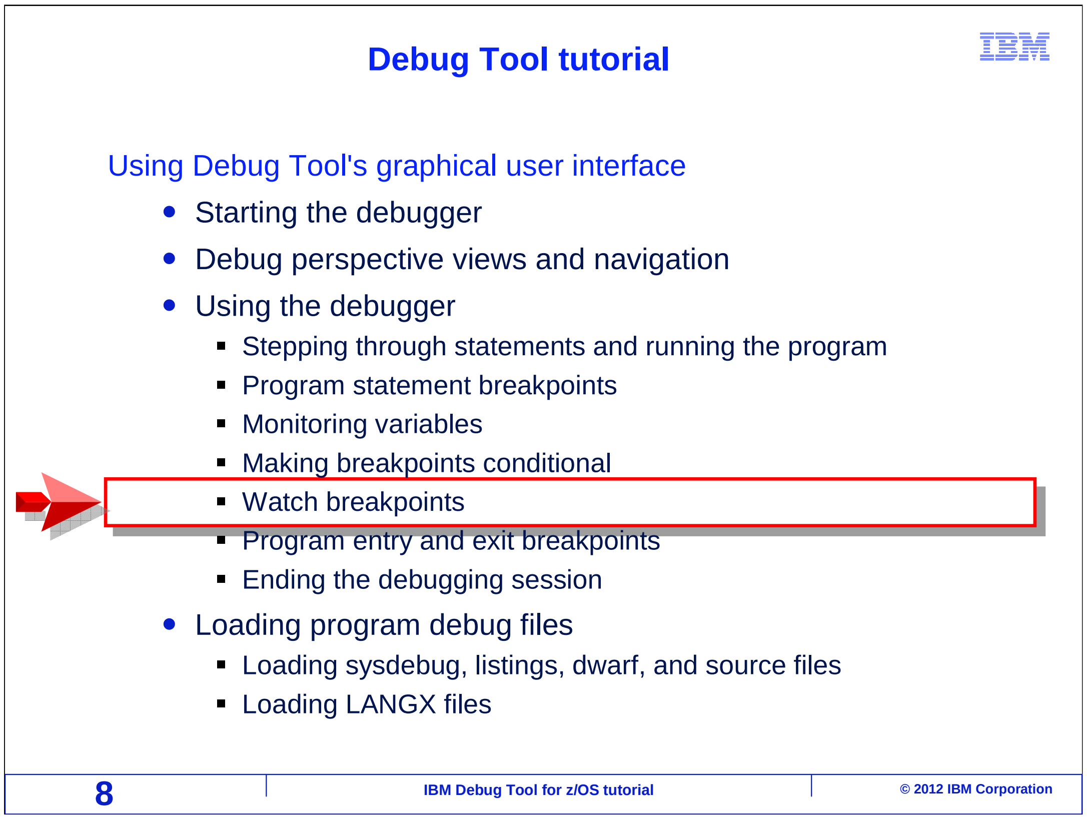
The breakpoint triggered the next time the program reached statement 404 and the condition was true

The program ran until it the next time that it reached statement 404 and the condition was true. Notice that CUST-ID, shown in the monitors view, has the right value to trigger the conditional breakpoint. In this example statement 404 ran several times when the condition was false, but it did not stop there until it was true.

Enter an expression to make a breakpoint conditional

- If an expression is entered, it is evaluated each time the breakpoint is encountered
  - The breakpoint triggers only if the condition is true
- Expression syntax:
  - Examples:
    - CUSTOMER-ID = '12345'
    - ACCOUNT-BALANCE > ‘1000’

Specifying an expression makes a breakpoint conditional. An expression has a variable name, an operator (<, >, =), and a comparator, which could be either a value or another variable.



Debug Tool tutorial

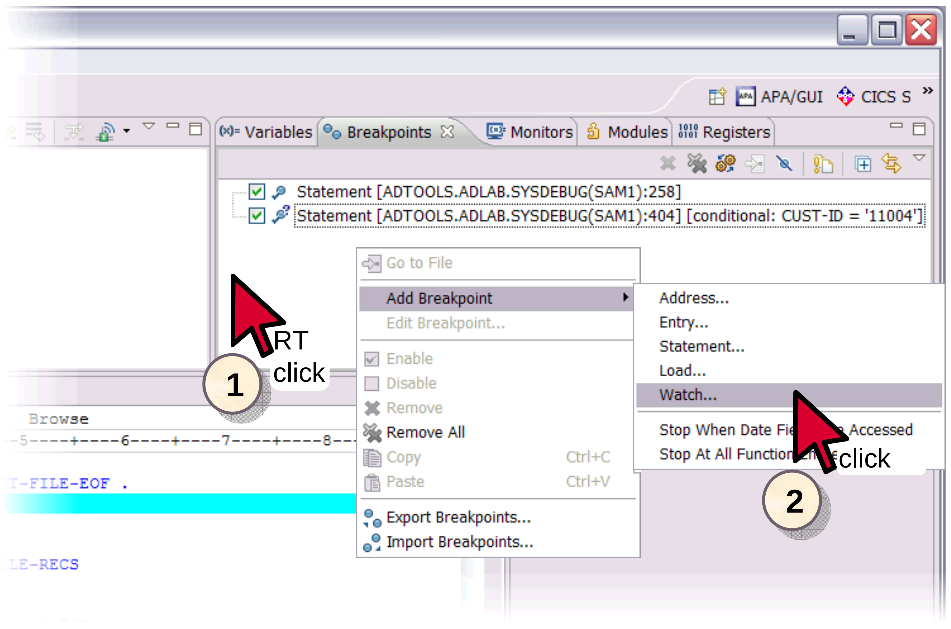
Using Debug Tool's graphical user interface

- Starting the debugger
- Debug perspective views and navigation
- Using the debugger
  - Stepping through statements and running the program
  - Program statement breakpoints
  - Monitoring variables
  - Making breakpoints conditional
  - Watch breakpoints
  - Program entry and exit breakpoints
  - Ending the debugging session
- Loading program debug files
  - Loading sysdebug, listings, dwarf, and source files
  - Loading LANGX files

8 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

Next, you will see how to set breakpoints that watch variables.

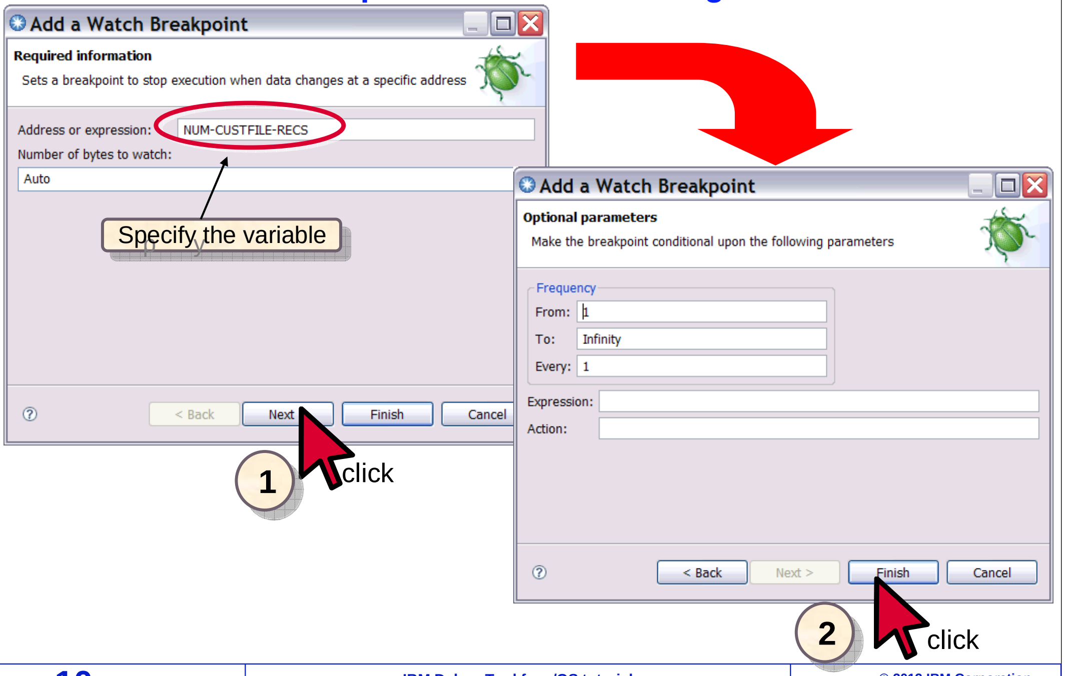
Create a “Watch” breakpoint



Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):404] [conditional: CUST-ID = '11004']

So far, you have seen how to set statement breakpoints. But there are other types of breakpoints. A "watch" breakpoint triggers when a variable's value changes. To add a watch breakpoint, right click in the breakpoints view, select “add breakpoint”, and then “watch”.

A Watch breakpoint triggers when a specified variable changes



Add a Watch Breakpoint

Required information

Sets a breakpoint to stop execution when data changes at a specific address

Address or expression: NUM-CUSTFILE-RECS

Number of bytes to watch: Auto

Add a Watch Breakpoint

Optional parameters

Make the breakpoint conditional upon the following parameters

Frequency

From: 1

To: Infinity

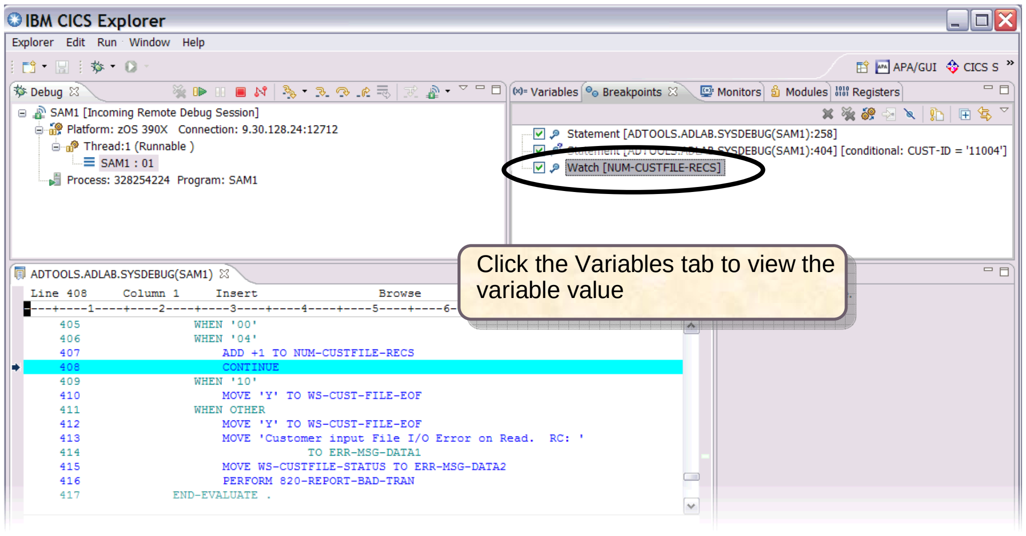
Every: 1

Expression:

Action:

The “Add a Watch Breakpoint” dialog is displayed. At the top of the first screen enter the variable or expression that is to be watched, and click Next. The second screen has optional parameters that allow you to specify a frequency and optionally make the watch breakpoint conditional by entering an expression. In this example, the frequency is not changed and no expression is entered. Click "finish".

Watch breakpoint for variable NUM-CUSTFILE-RECS has been added


IBM CICS Explorer
Explorer Edit Run Window Help
APA/GUI CICS S
Debug
Variables Breakpoints Monitors Modules Registers
Click the Variables tab to view the variable value

SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:12712
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

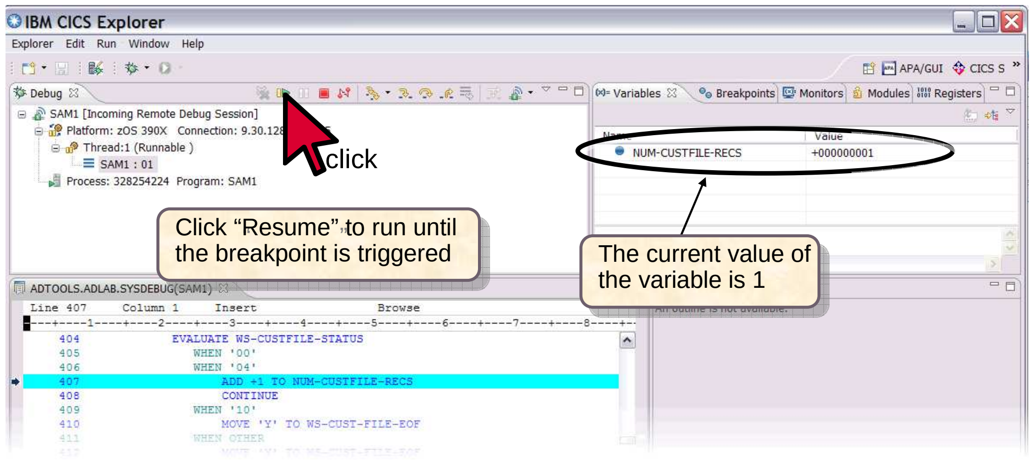
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):404] [conditional: CUST-ID = '11004']
Watch [NUM-CUSTFILE-RECS]

ADTOOLS.ADLAB.SYSDEBUG(SAM1)

```
Line 408      Column 1      Insert                   Browse
----+----1----+----2----+----3----+----4----+----5----+----6-
                   WHEN '00'
                   WHEN '04'
                       ADD +1 TO NUM-CUSTFILE-RECS
                       CONTINUE
                   WHEN '10'
                       MOVE 'Y' TO WS-CUST-FILE-EOF
                   WHEN OTHER
                       MOVE 'Y' TO WS-CUST-FILE-EOF
                       MOVE 'Customer input File I/O Error on Read.  RC: '
                                   TO ERR-MSG-DATA1
                       MOVE WS-CUSTFILE-STATUS TO ERR-MSG-DATA2
                       PERFORM 820-REPORT-BAD-TRAN
                END-EVALUATE .
```

The watch breakpoint was added, and is displayed in the breakpoints view.

Run the program


IBM CICS Explorer
Explorer Edit Run Window Help
Debug
Variables
Breakpoints
Monitors
Modules
Registers
click
Click “Resume” to run until the breakpoint is triggered
The current value of the variable is 1
ADTOOLS.ADLAB.SYSDEBUG(SAM1)

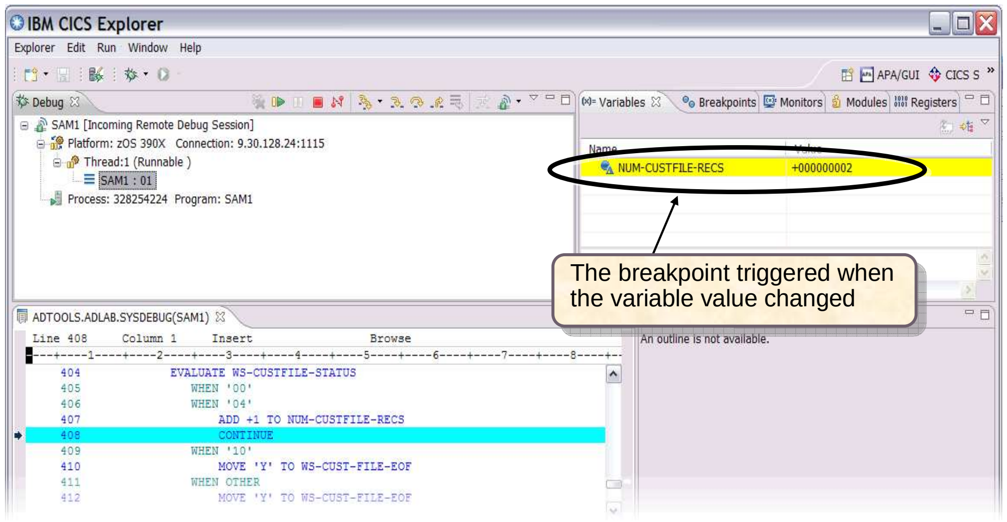
SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

Name	Value
NUM-CUSTFILE-RECS	+000000001

```
Line 407     Column 1      Insert                   Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
           EVALUATE WS-CUSTFILE-STATUS
              WHEN '00'
              WHEN '04'
                  ADD +1 TO NUM-CUSTFILE-RECS
                  CONTINUE
              WHEN '10'
                  MOVE 'Y' TO WS-CUST-FILE-EOF
              WHEN OTHER
```

Here, the variables view is selected, which is displaying the watched variable. The resume button is clicked.

The watch breakpoint triggered when the variable changed


IBM CICS Explorer
Explorer Edit Run Window Help
APA/GUI CICS S
Debug
Variables Breakpoints Monitors Modules Registers
Name Value
The breakpoint triggered when the variable value changed
An outline is not available.

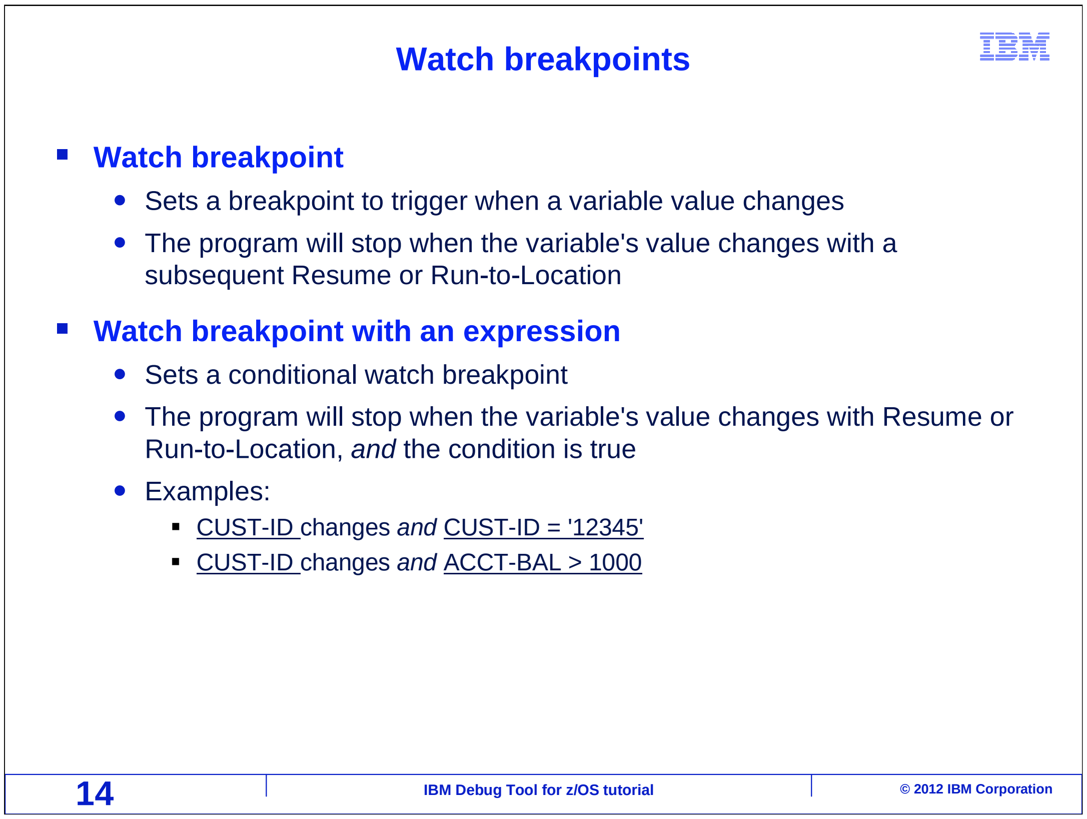
SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:1115
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

Name	Value
NUM-CUSTFILE-RECS	+000000002

ADTOOLS.ADLAB.SYSDEBUG(SAM1)

```
Line 408     Column 1     Insert                Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
            EVALUATE WS-CUSTFILE-STATUS
               WHEN '00'
               WHEN '04'
                   ADD +1 TO NUM-CUSTFILE-RECS
                   CONTINUE
               WHEN '10'
                   MOVE 'Y' TO WS-CUST-FILE-EOF
               WHEN OTHER
                   MOVE 'Y' TO WS-CUST-FILE-EOF
```

The watch breakpoint triggered. Notice that the value of the variable changed, and that the program is stopped at the next statement to run after the statement that changed the variable.

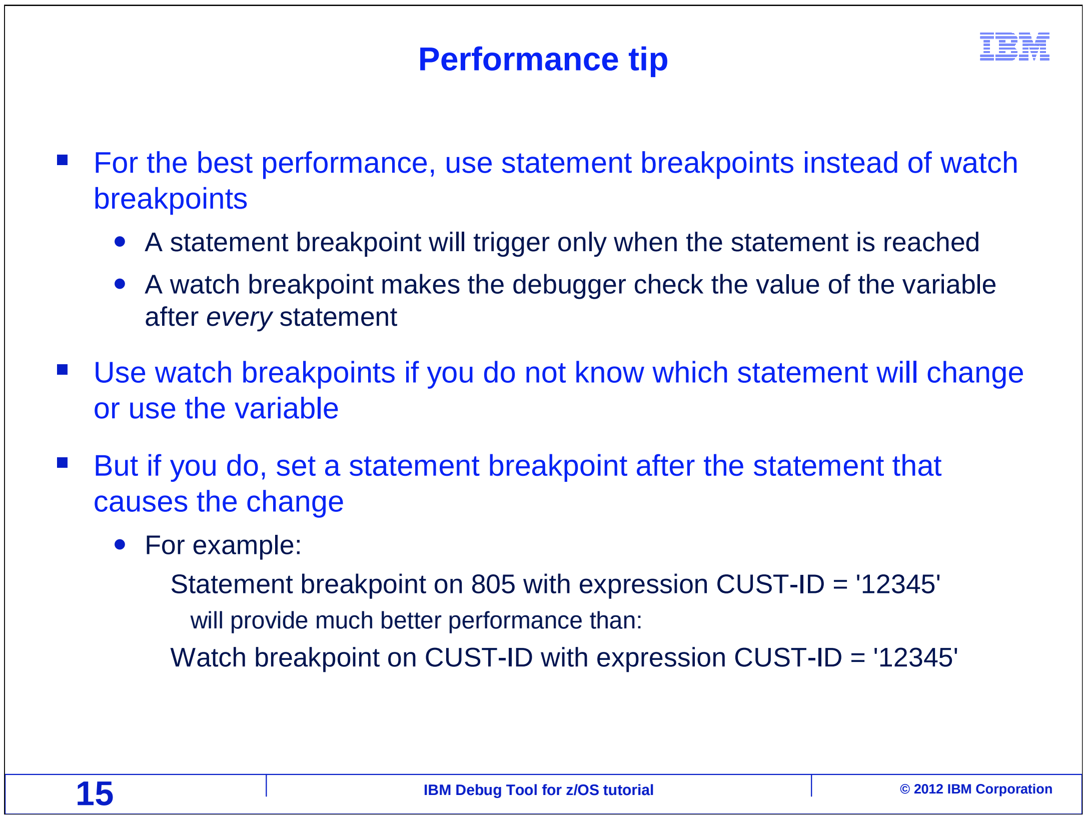


Watch breakpoints

- **Watch breakpoint**
  - Sets a breakpoint to trigger when a variable value changes
  - The program will stop when the variable's value changes with a subsequent Resume or Run-to-Location
- **Watch breakpoint with an expression**
  - Sets a conditional watch breakpoint
  - The program will stop when the variable's value changes with Resume or Run-to-Location, *and* the condition is true
  - Examples:
    - CUST-ID changes *and* CUST-ID = '12345'
    - CUST-ID changes *and* ACCT-BAL > 1000

14 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

A watch breakpoint will trigger when a variable value changes. Optionally, watch breakpoints can be made conditional by specifying an expression.


Performance tip

- For the best performance, use statement breakpoints instead of watch breakpoints
  - A statement breakpoint will trigger only when the statement is reached
  - A watch breakpoint makes the debugger check the value of the variable after *every* statement
- Use watch breakpoints if you do not know which statement will change or use the variable
- But if you do, set a statement breakpoint after the statement that causes the change
  - For example:

    Statement breakpoint on 805 with expression CUST-ID = '12345'

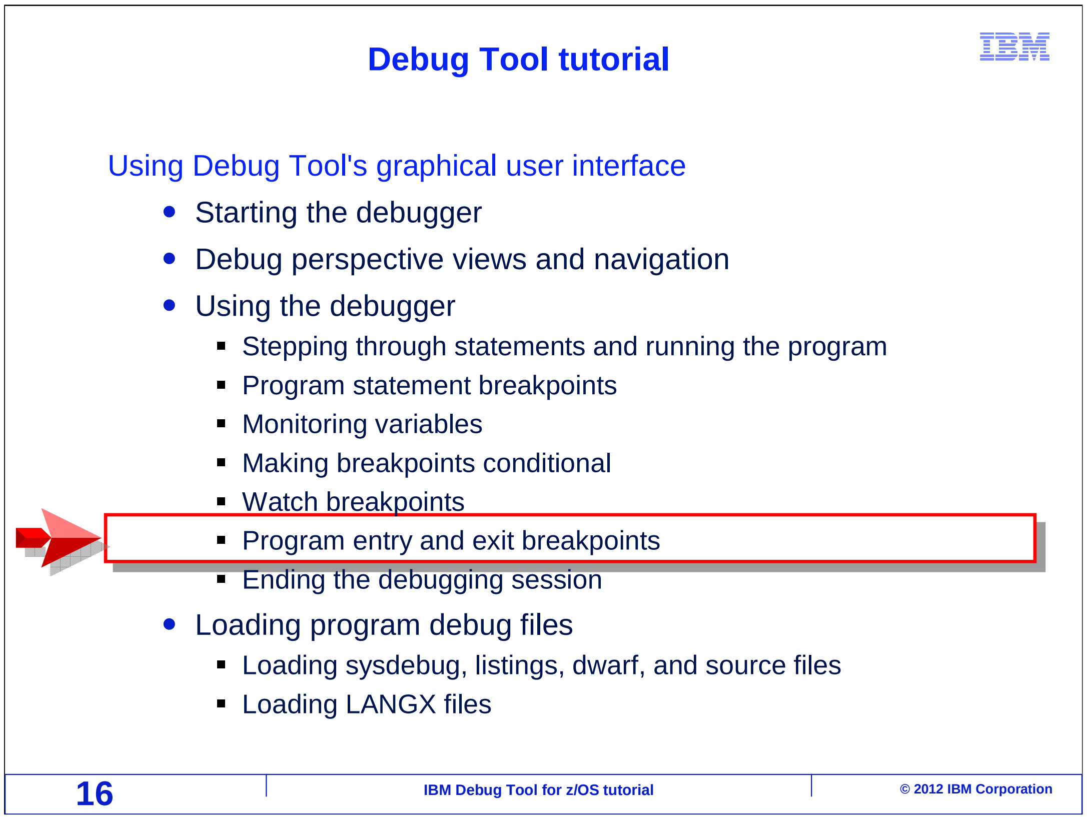
    will provide much better performance than:

    Watch breakpoint on CUST-ID with expression CUST-ID = '12345'

15 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

When a watch breakpoint is set, the debugger must check the value of the watched variable after every statement runs. Keep this in mind when setting breakpoints, because it affects the performance of the debugger. If you have long running programs, and want to optimize the debugger's performance, there may be a better performing alternative.

If you know where in a program a target variable will be used, then you can set a statement breakpoint instead. Specify a conditional expression to check the value of the variable you want to watch at the right statement. Then the program will stop at the statement when the variable has a target value. This performs much better, since the debugger only checks for the condition when the specific statement runs, rather than checking if the variable changed after every statement.



Debug Tool tutorial

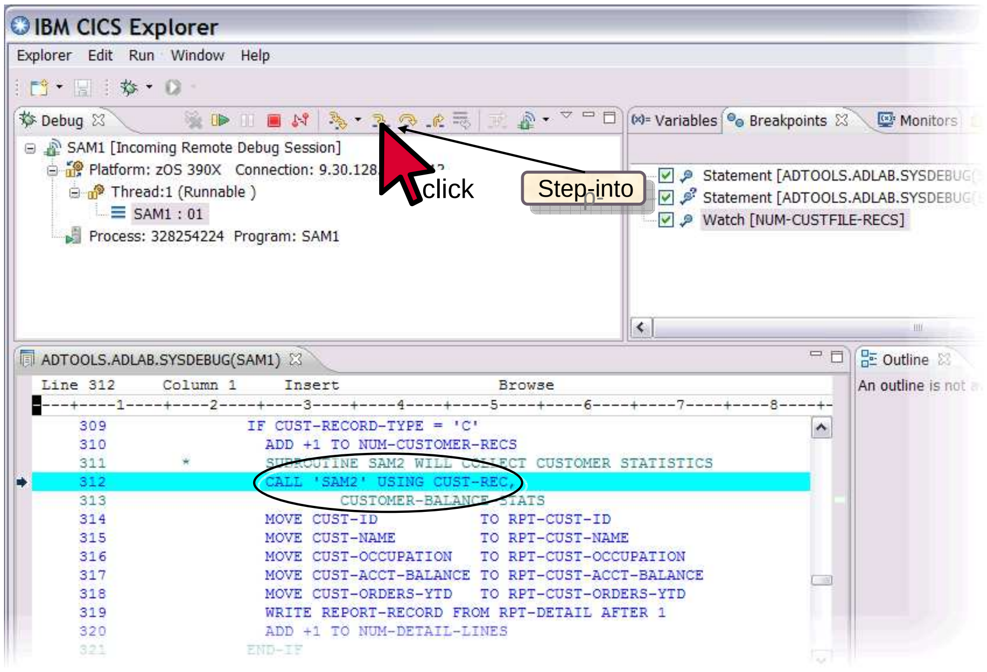
Using Debug Tool's graphical user interface

- Starting the debugger
- Debug perspective views and navigation
- Using the debugger
  - Stepping through statements and running the program
  - Program statement breakpoints
  - Monitoring variables
  - Making breakpoints conditional
  - Watch breakpoints
  - Program entry and exit breakpoints
  - Ending the debugging session
- Loading program debug files
  - Loading sysdebug, listings, dwarf, and source files
  - Loading LANGX files

16 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

So far, you have seen how to set statement and watch breakpoints. Next, you will see how to set breakpoints that trigger when a specific program or subprogram is entered.

You can step into a subprogram or function



SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

Statement [ADTOOLS.ADLAB.SYSDEBUG
Statement [ADTOOLS.ADLAB.SYSDEBUG
Watch [NUM-CUSTFILE-RECS]

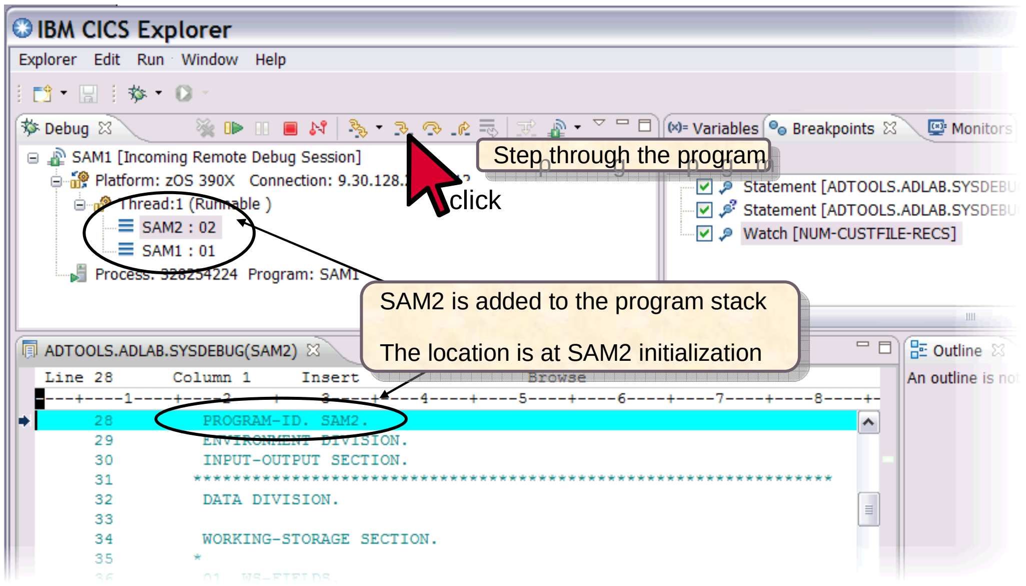
ADTOOLS.ADLAB.SYSDEBUG(SAM1)

```
Line 312     Column 1     Insert                 Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
    309              IF CUST-RECORD-TYPE = 'C'
    310                ADD +1 TO NUM-CUSTOMER-RECS
    311         *      SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
    312                CALL 'SAM2' USING CUST-REC,
    313                        CUSTOMER-BALANCE-STATS
    314                MOVE CUST-ID             TO RPT-CUST-ID
    315                MOVE CUST-NAME           TO RPT-CUST-NAME
    316                MOVE CUST-OCCUPATION     TO RPT-CUST-OCCUPATION
    317                MOVE CUST-ACCT-BALANCE   TO RPT-CUST-ACCT-BALANCE
    318                MOVE CUST-ORDERS-YTD     TO RPT-CUST-ORDERS-YTD
    319                WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
    320                ADD +1 TO NUM-DETAIL-LINES
    321              END-IF
```

But first, here is a simple example of how you can follow the logic of an application from one program to another. A program named SAM1 is running. You can tell because the program name is highlighted in the debug view, and it is also displayed in the tab of the source view.

The next statement to run is line 312, which is a "CALL" statement that will pass control to another program named SAM2. It has not yet executed the CALL statement. To follow the logic into the subprogram, click “Step-into”.

After stepping into a subprogram


IBM CICS Explorer
Explorer Edit Run Window Help
Debug
Variables
Breakpoints
Monitors
Step through the program
click
SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.
Thread:1 (Runnable)
SAM2 : 02
SAM1 : 01
Process: 328254224 Program: SAM1
Statement [ADTOOLS.ADLAB.SYSDEBU
Statement [ADTOOLS.ADLAB.SYSDEBU
Watch [NUM-CUSTFILE-RECS]
SAM2 is added to the program stack
The location is at SAM2 initialization
ADTOOLS.ADLAB.SYSDEBUG(SAM2)
Outline
An outline is no
Line 28 Column 1 Insert Browse

```
       PROGRAM-ID. SAM2.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
      ******************************************************************
       DATA DIVISION.

       WORKING-STORAGE SECTION.
      *
```

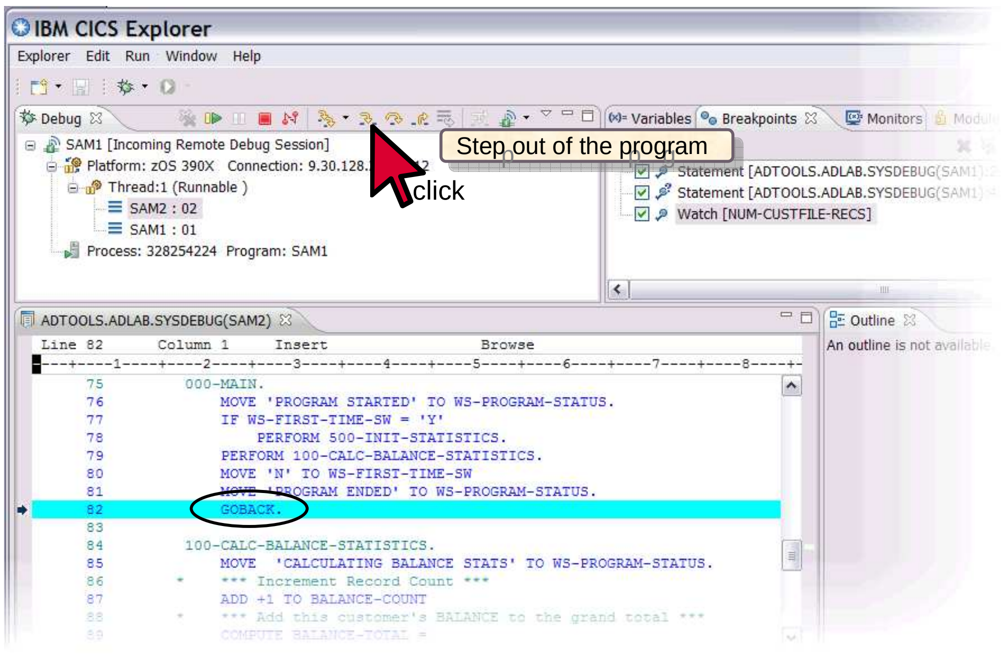
That stepped into the subprogram, SAM2.

There are two cases when you are sitting on a "CALL" statement and step. Either the subprogram has been compiled for use with the debugger, or not. For example, in the case of an Enterprise COBOL subprogram, if it is compiled with the "TEST" compiler option, then it is compiled for the debugger. If it has been compiled with "NOTEST", then it is not.

In this example, the subprogram is compiled for use with the debugger, so it steps in. If this program had not been compiled for the debugger, then "STEP" would run the subprogram, but the debugger would not display it. The debugger would pause at the next statement after the "CALL" statement in the higher level program.

"Step-into" is clicked several times to step through the program.

You can STEP out of a subprogram or function


IBM CICS Explorer
Explorer Edit Run Window Help
Debug
Variables Breakpoints Monitors
Step out of the program
click
ADTOOLS.ADLAB.SYSDEBUG(SAM2)
Outline
An outline is not available

```
SAM1 [Incoming Remote Debug Session]
  Platform: zOS 390X  Connection: 9.30.128.
    Thread:1 (Runnable )
      SAM2 : 02
      SAM1 : 01
  Process: 328254224  Program: SAM1
```

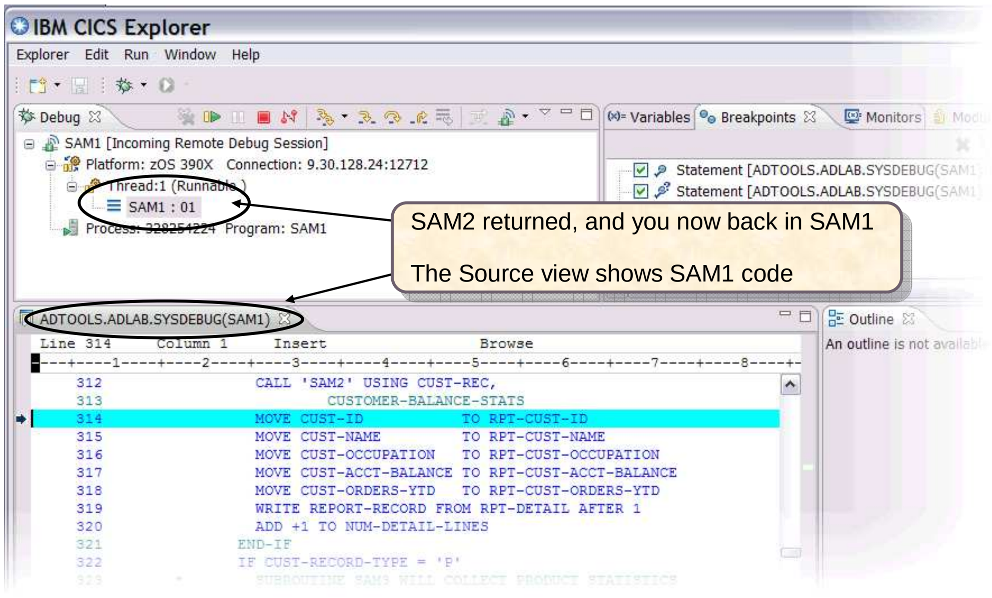
```
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1)
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1)
Watch [NUM-CUSTFILE-RECS]
```

```
Line 82      Column 1      Insert                   Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
        000-MAIN.
            MOVE 'PROGRAM STARTED' TO WS-PROGRAM-STATUS.
            IF WS-FIRST-TIME-SW = 'Y'
                PERFORM 500-INIT-STATISTICS.
            PERFORM 100-CALC-BALANCE-STATISTICS.
            MOVE 'N' TO WS-FIRST-TIME-SW
            MOVE 'PROGRAM ENDED' TO WS-PROGRAM-STATUS.
            GOBACK.

        100-CALC-BALANCE-STATISTICS.
            MOVE  'CALCULATING BALANCE STATS' TO WS-PROGRAM-STATUS.
       *    *** Increment Record Count ***
            ADD +1 TO BALANCE-COUNT
       *    *** Add this customer's BALANCE to the grand total ***
            COMPUTE BALANCE-TOTAL =
```

After stepping or running through the program, the last statement in the subprogram is reached. "Step-into" is clicked again.

Returned to the calling program



SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:12712
Thread:1 (Runnable)
SAM1 : 01
Process: 328254224 Program: SAM1

Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1

SAM2 returned, and you now back in SAM1

The Source view shows SAM1 code

ADTOOLS.ADLAB.SYSDEBUG(SAM1)

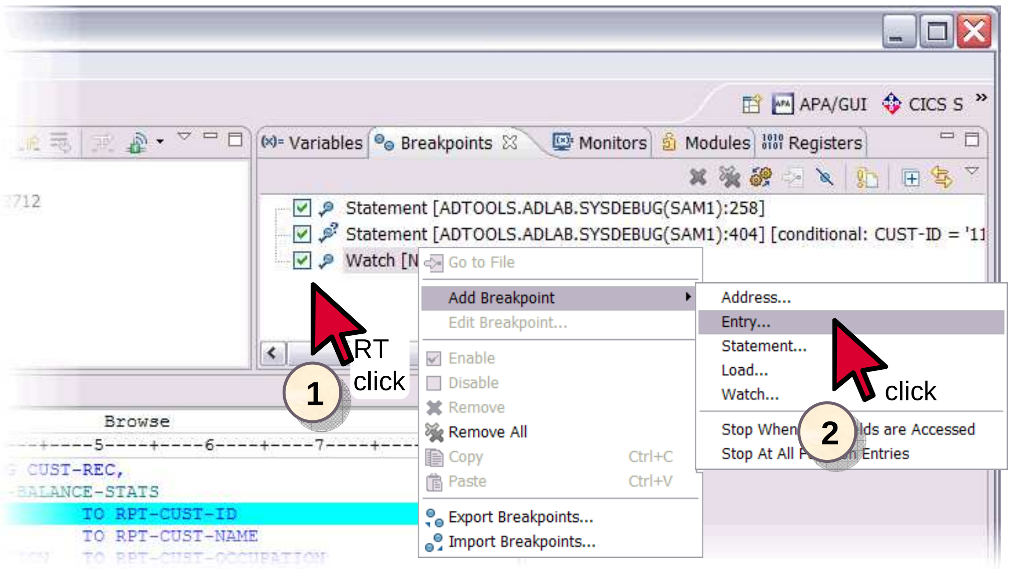
An outline is not available

```
Line 314      Column 1       Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
                     CALL 'SAM2' USING CUST-REC,
                             CUSTOMER-BALANCE-STATS
                     MOVE CUST-ID            TO RPT-CUST-ID
                     MOVE CUST-NAME          TO RPT-CUST-NAME
                     MOVE CUST-OCCUPATION    TO RPT-CUST-OCCUPATION
                     MOVE CUST-ACCT-BALANCE  TO RPT-CUST-ACCT-BALANCE
                     MOVE CUST-ORDERS-YTD    TO RPT-CUST-ORDERS-YTD
                     WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
                     ADD +1 TO NUM-DETAIL-LINES
                   END-IF
                   IF CUST-RECORD-TYPE = 'P'
      *              SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
```

IBM Debug Tool for z/OS tutorial

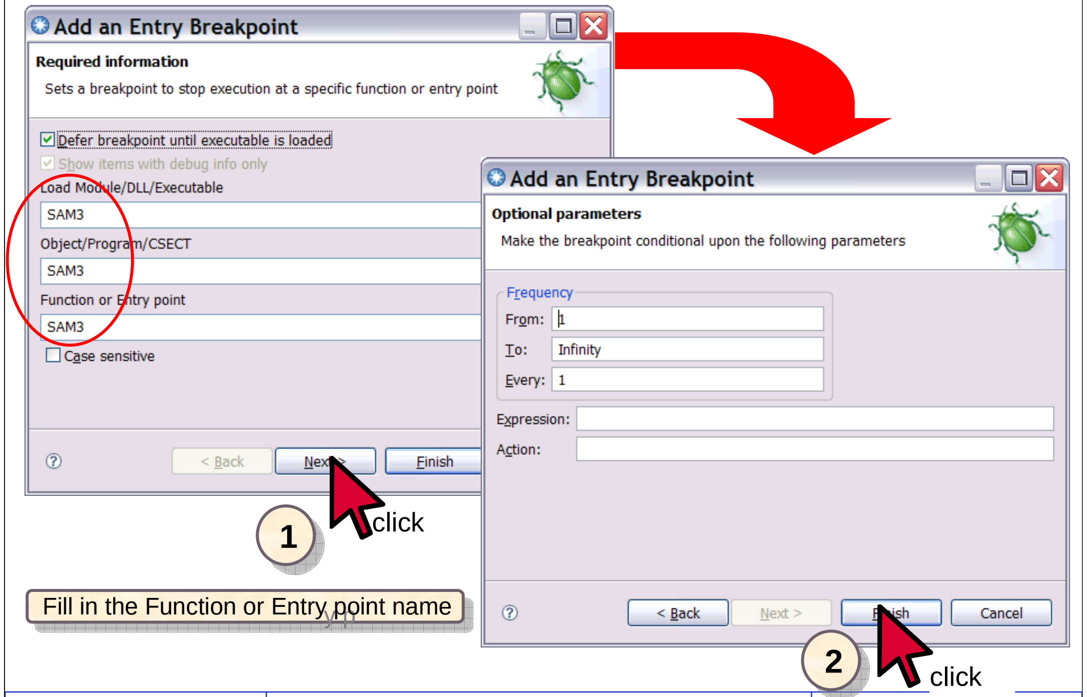
After stepping, control is returned back to the higher level program. The debugger paused at the next logical statement, which is the statement after the CALL. Now you have seen one way to follow logic into a subprogram, step in and step out.

Create an entry breakpoint



But it can be cumbersome to have to step into every subprogram that you want to debug. So there is a simpler way to pause when a specific subprogram in entered – an "entry" breakpoint. To set an entry breakpoint, right click in the Breakpoints view, and select “Add Breakpoint” and then “Entry”.

Create an entry breakpoint



Add an Entry Breakpoint

Required information

Sets a breakpoint to stop execution at a specific function or entry point

☑ Defer breakpoint until executable is loaded

☑ Show items with debug info only

Load Module/DLL/Executable: SAM3

Object/Program/CSECT: SAM3

Function or Entry point: SAM3

☐ Case sensitive

< Back | Next > | Finish

Add an Entry Breakpoint

Optional parameters

Make the breakpoint conditional upon the following parameters

Frequency

From: 1

To: Infinity

Every: 1

Expression:

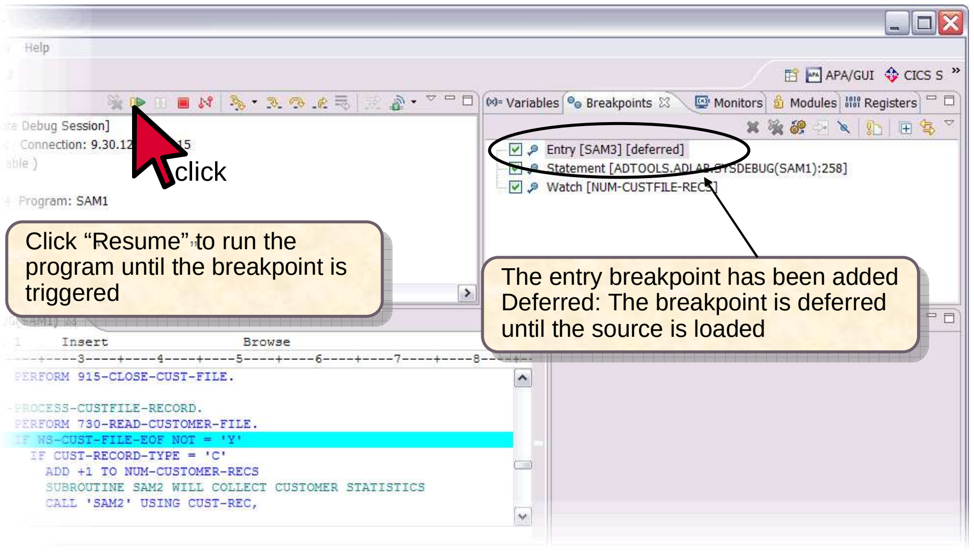
Action:

< Back | Next > | Finish | Cancel

The “Add an Entry Breakpoint” pop-up appears. If the sub program has not been called yet, then it may not be loaded in memory. If that is the case, click the check box labeled “Defer breakpoint until executable is loaded”. If you are not sure, check the box anyway. The breakpoint will still trigger if the module already happens to be loaded. Enter the full names of the load module, CSECT or program name, and entry point name, and click "next".

On the next page, you can optionally specify a frequency or an expression to make the breakpoint conditional. Click "finish".

Create an entry breakpoint


Help
APA/GUI
CICS S
Variables
Breakpoints
Monitors
Modules
Registers
click
Click “Resume” to run the program until the breakpoint is triggered
The entry breakpoint has been added
Deferred: The breakpoint is deferred until the source is loaded

Debug Session]
Connection: 9.30.12 15
Program: SAM1

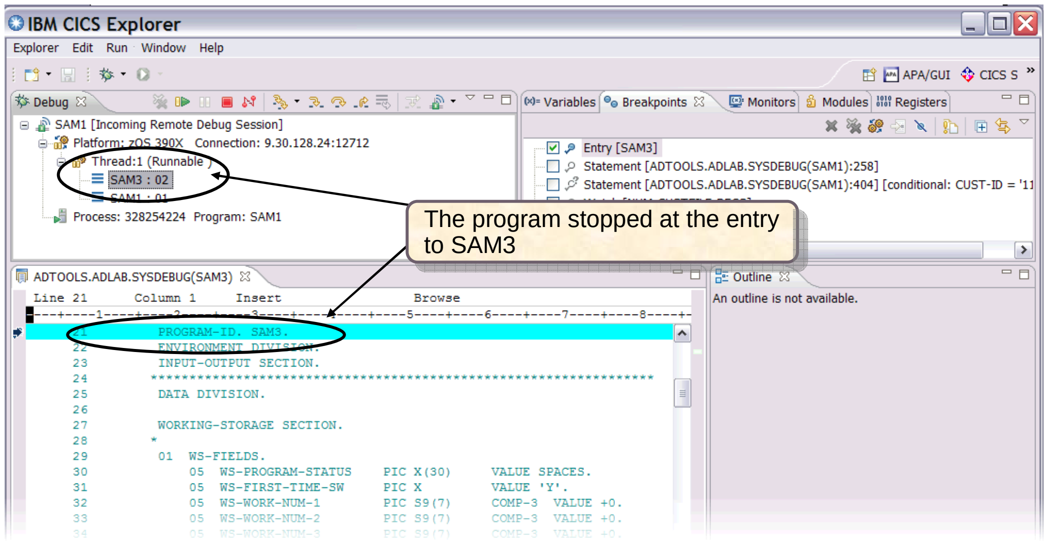
Entry [SAM3] [deferred]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]
Watch [NUM-CUSTFILE-RECS]

```
      Insert                    Browse
----+----3----+----4----+----5----+----6----+----7----+----8----+--
PERFORM 915-CLOSE-CUST-FILE.

PROCESS-CUSTFILE-RECORD.
PERFORM 730-READ-CUSTOMER-FILE.
IF WS-CUST-FILE-EOF NOT = 'Y'
  IF CUST-RECORD-TYPE = 'C'
    ADD +1 TO NUM-CUSTOMER-RECS
    SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
    CALL 'SAM2' USING CUST-REC,
```

The breakpoint is added, and appears in the breakpoints view. Click “resume” to" run the program.

After stopping at the entry breakpoint


IBM CICS Explorer
Explorer Edit Run Window Help
APA/GUI CICS S
Debug
Variables Breakpoints Monitors Modules Registers
Outline
The program stopped at the entry to SAM3

SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:12712
Thread:1 (Runnable)
SAM3 : 02
SAM1 : 01
Process: 328254224 Program: SAM1

Entry [SAM3]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):258]
Statement [ADTOOLS.ADLAB.SYSDEBUG(SAM1):404] [conditional: CUST-ID = '11

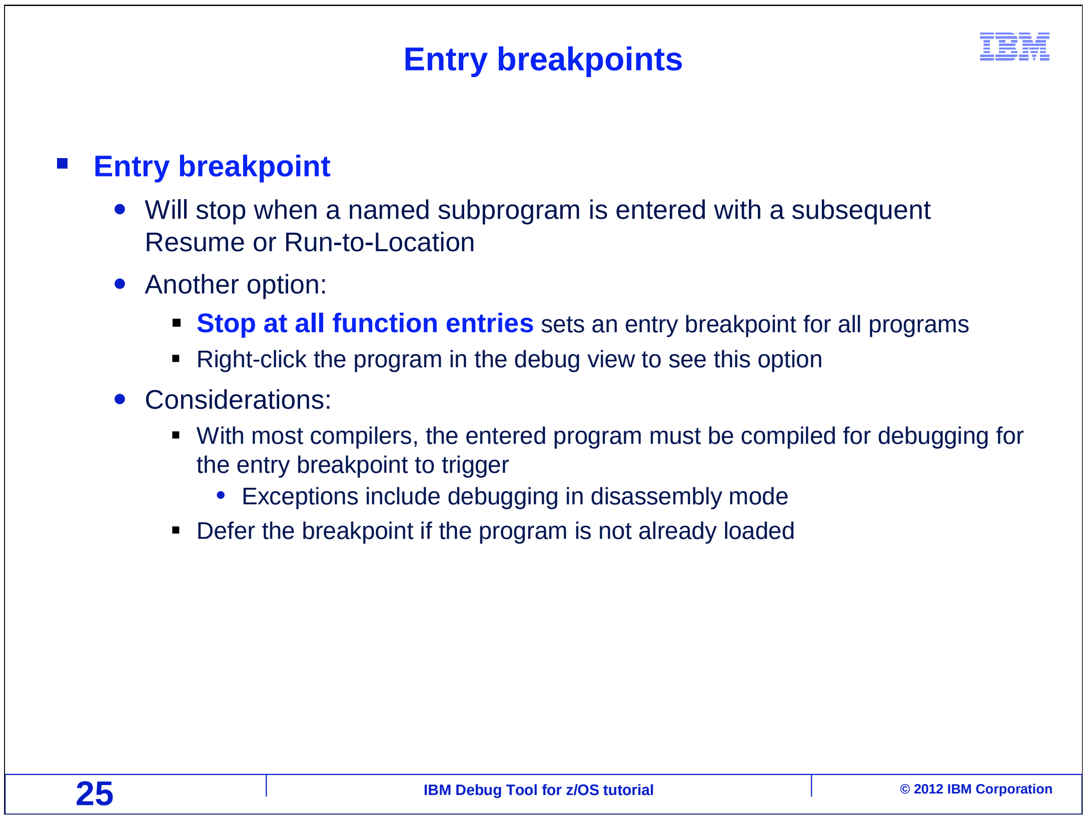
An outline is not available.

ADTOOLS.ADLAB.SYSDEBUG(SAM3)

```
Line 21       Column 1     Insert                Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
            PROGRAM-ID. SAM3.
            ENVIRONMENT DIVISION.
            INPUT-OUTPUT SECTION.
           ****************************************************************
            DATA DIVISION.

            WORKING-STORAGE SECTION.
           *
            01  WS-FIELDS.
                05  WS-PROGRAM-STATUS     PIC X(30)     VALUE SPACES.
                05  WS-FIRST-TIME-SW      PIC X         VALUE 'Y'.
                05  WS-WORK-NUM-1         PIC S9(7)     COMP-3  VALUE +0.
                05  WS-WORK-NUM-2         PIC S9(7)     COMP-3  VALUE +0.
                05  WS-WORK-NUM-3         PIC S9(7)     COMP-3  VALUE +0.
```

The breakpoint triggered when the program was entered. It is paused the at the entry, and you can follow the logic by stepping in.

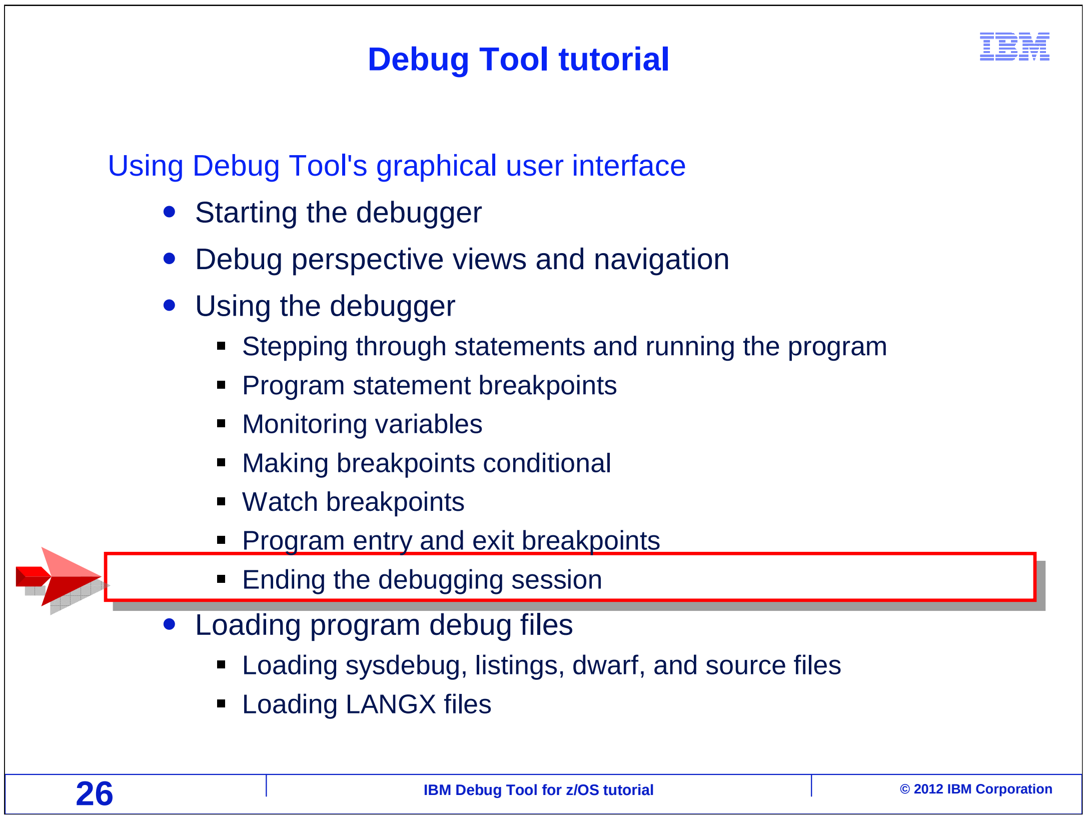


Entry breakpoints

- **Entry breakpoint**
  - Will stop when a named subprogram is entered with a subsequent Resume or Run-to-Location
  - Another option:
    - **Stop at all function entries** sets an entry breakpoint for all programs
    - Right-click the program in the debug view to see this option
  - Considerations:
    - With most compilers, the entered program must be compiled for debugging for the entry breakpoint to trigger
      - Exceptions include debugging in disassembly mode
    - Defer the breakpoint if the program is not already loaded

25 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

An entry breakpoint gives you an easy way to run the application until it reaches a specific program. With most compilers, the subprogram must be compiled for debugging for the entry breakpoint to trigger. Do not forget to click the “Defer breakpoint until executable is loaded” check box when setting the breakpoint, if you are not sure that the module is already in memory.



Debug Tool tutorial

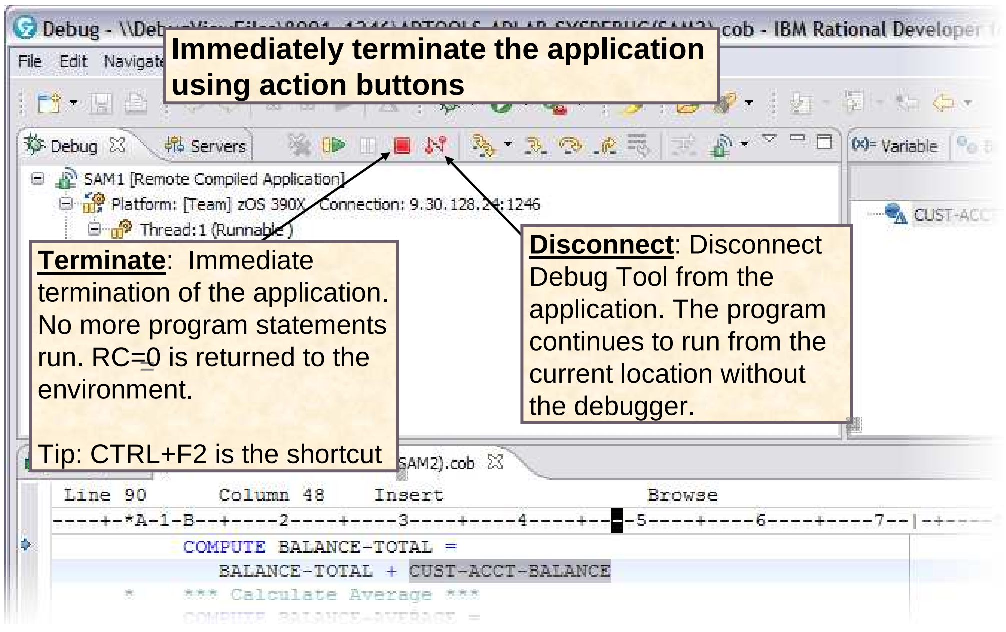
Using Debug Tool's graphical user interface

- Starting the debugger
- Debug perspective views and navigation
- Using the debugger
  - Stepping through statements and running the program
  - Program statement breakpoints
  - Monitoring variables
  - Making breakpoints conditional
  - Watch breakpoints
  - Program entry and exit breakpoints
  - Ending the debugging session
- Loading program debug files
  - Loading sysdebug, listings, dwarf, and source files
  - Loading LANGX files

26 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

Next, you will see options available to end a debugging session.

Termination action buttons



Immediately terminate the application using action buttons

Terminate: Immediate termination of the application. No more program statements run. RC=0 is returned to the environment.

Tip: CTRL+F2 is the shortcut

Disconnect: Disconnect Debug Tool from the application. The program continues to run from the current location without the debugger.

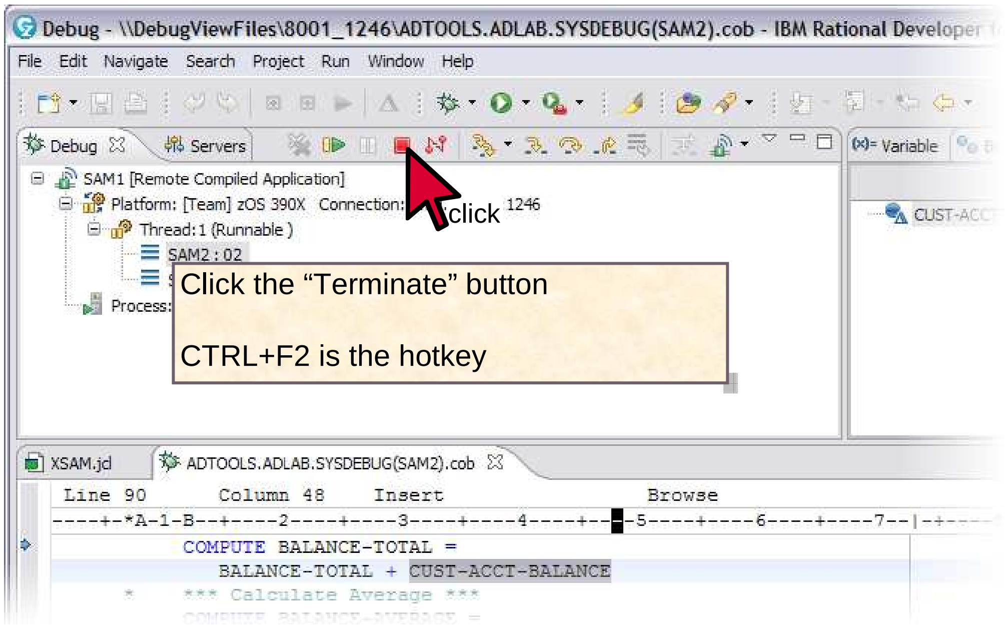
```
SAM1 [Remote Compiled Application]
  Platform: [Team] zOS 390X  Connection: 9.30.128.24:1246
    Thread:1 (Runnable)
```

```
 Line 90      Column 48    Insert                  Browse
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----
              COMPUTE BALANCE-TOTAL =
                 BALANCE-TOTAL + CUST-ACCT-BALANCE
       *      *** Calculate Average ***
              COMPUTE BALANCE-AVERAGE =
```

When you click "resume", the application runs until the next breakpoint is triggered. However, if the program does not reach or trigger any breakpoints, it will run to termination. That is one way to end your debugging session, just run the program until it finishes.

However, you can end a debugging session at any time with the terminate or disconnect buttons. "Terminate" stops the application immediately. No more statements run, and the application is given a zero return code. "Disconnect" ends the debugging session, but allows the application to run normally, continuing from the current statement.

Click the “Terminate” button



Debug - \\DebugViewFiles\8001_1246\ADTOOLS.ADLAB.SYSDEBUG(SAM2).cob - IBM Rational Developer
File Edit Navigate Search Project Run Window Help
Debug
Servers
Variable
SAM1 [Remote Compiled Application]
Platform: [Team] zOS 390X Connection:
1246
Thread:1 (Runnable)
SAM2 : 02
Process:
click
CUST-ACC
XSAM.jcl
ADTOOLS.ADLAB.SYSDEBUG(SAM2).cob

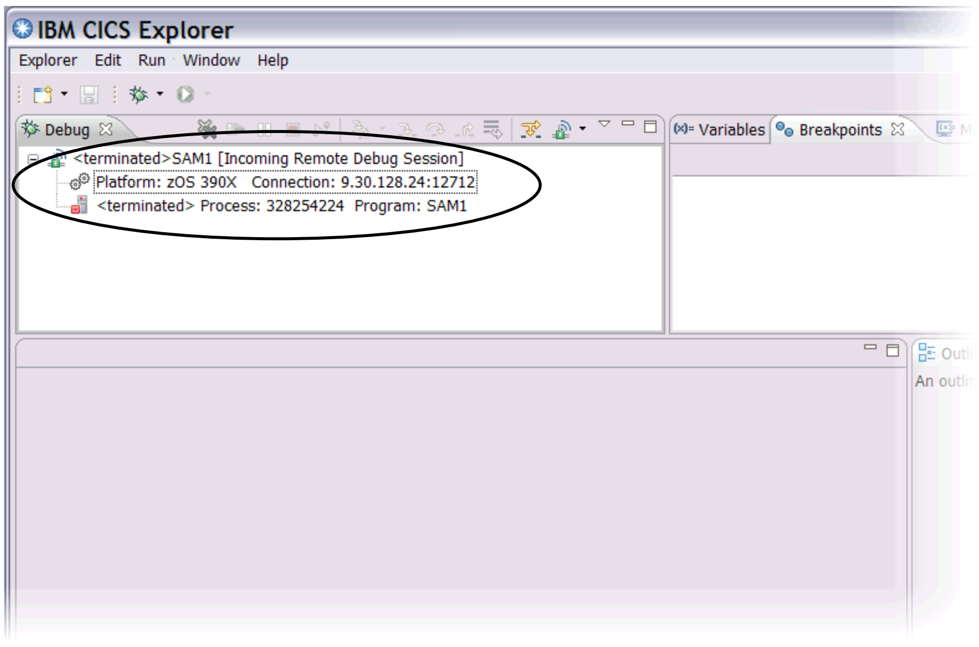
Click the “Terminate” button

CTRL+F2 is the hotkey

```
 Line 90      Column 48     Insert                       Browse
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----
            COMPUTE BALANCE-TOTAL =
               BALANCE-TOTAL + CUST-ACCT-BALANCE
      *     *** Calculate Average ***
            COMPUTE BALANCE-AVERAGE =
```

The “terminate” button is clicked.

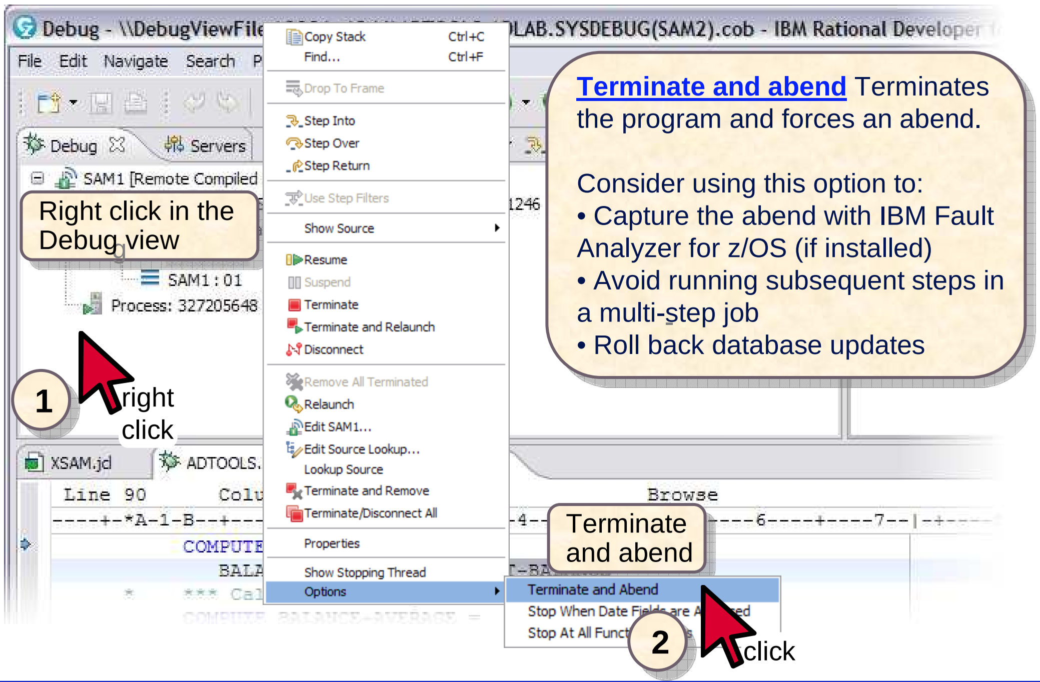
The debugging session ended



<terminated>SAM1 [Incoming Remote Debug Session]
Platform: zOS 390X Connection: 9.30.128.24:12712
<terminated> Process: 328254224 Program: SAM1

The application was terminated immediately, and the debugging session ended. The debugger is now ready for the next debugging session to start, if needed.

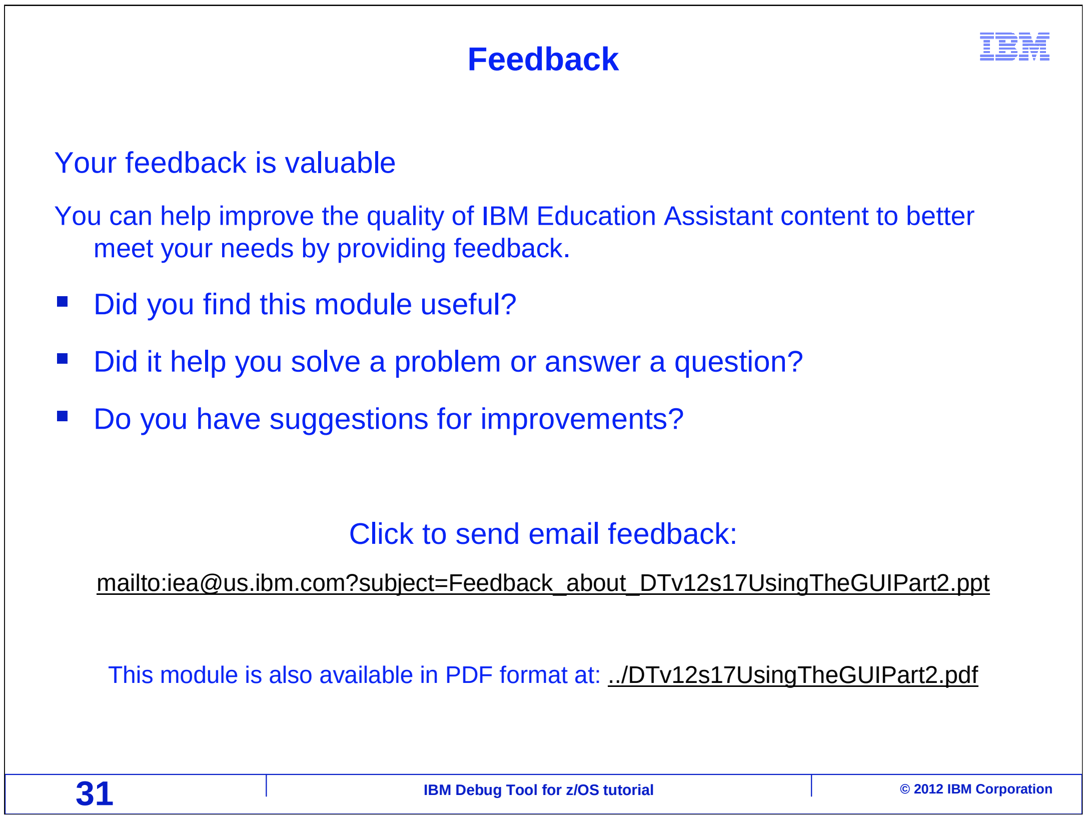
Force an immediate termination with abend



Terminate and abend Terminates the program and forces an abend.

Consider using this option to:
- Capture the abend with IBM Fault Analyzer for z/OS (if installed)
- Avoid running subsequent steps in a multi-step job
- Roll back database updates

Another way to end a session is to right click in the white space of the debug view, and select “Options” and then “Terminate and Abend”. This forces the program to terminate with an abend. Consider doing this if you want the system to collect a dump. If you have IBM’s Fault Analyzer product, perhaps you plan to use it to see a detailed analysis. In some cases, forcing an abend can prevent subsequent steps in a batch job from running. And in some applications an abend will force changes made to databases to be rolled back.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

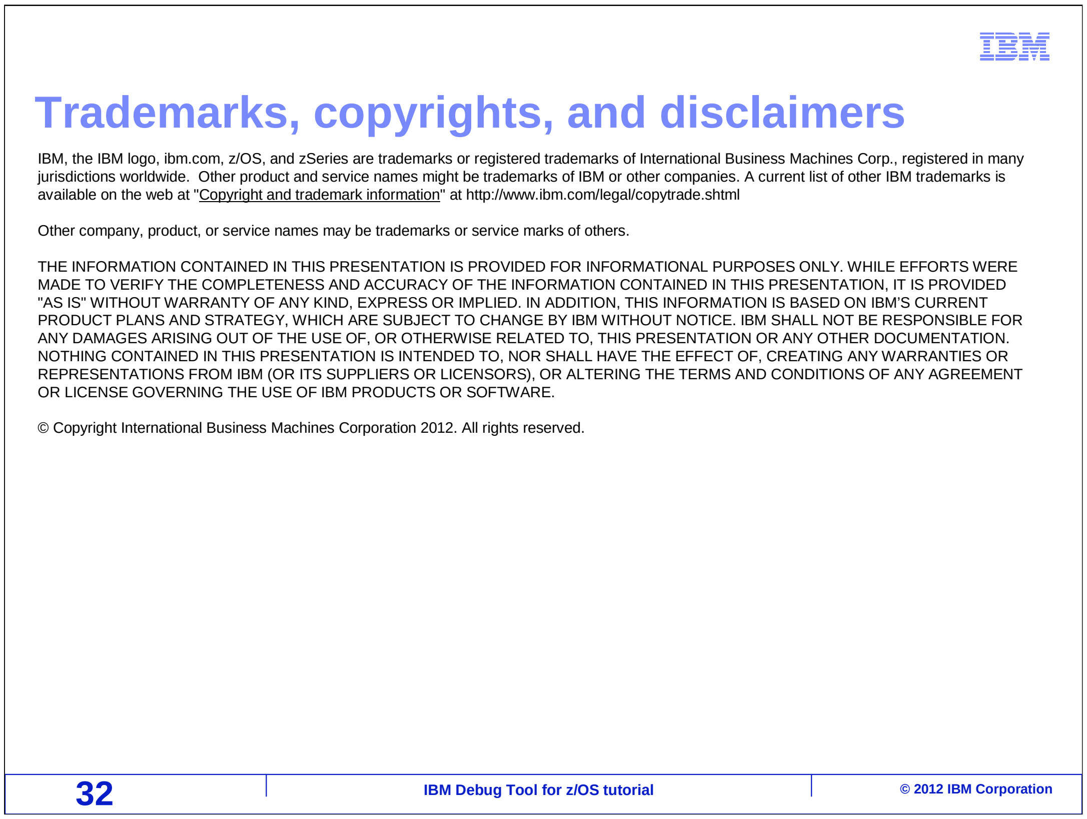
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_DTv12s17UsingTheGUIPart2.ppt

This module is also available in PDF format at: ../DTv12s17UsingTheGUIPart2.pdf

31 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers