WebSphere® WebSphere Developer Debugger for zSeries

IBM

Version 6.0.1

Debugging MVS COBOL with calls to a C program and C++ DLL

WebSphere® WebSphere Developer Debugger for zSeries

Version 6.0.1

Debugging MVS COBOL with calls to a C program and C++ DLL

Note

Before using this information and the product it supports, read the information in Notices at the end of this book.

© Copyright International Business Machines Corporation 2005. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Debugging MVS COBOL with calls to a C program and C++ DLL

Objectives

This tutorial will guide you through the launch of a debug session for a sample MVS™ COBOL application which calls a separately-compiled C subroutine. Parameters are passed by reference. The C module, in turn, calls two entry points in a C++ DLL.

Description

To debug the application, complete the following steps:

1. Prepare to debug.
2. Get accustomed to the user interface.
3. Debug the application.

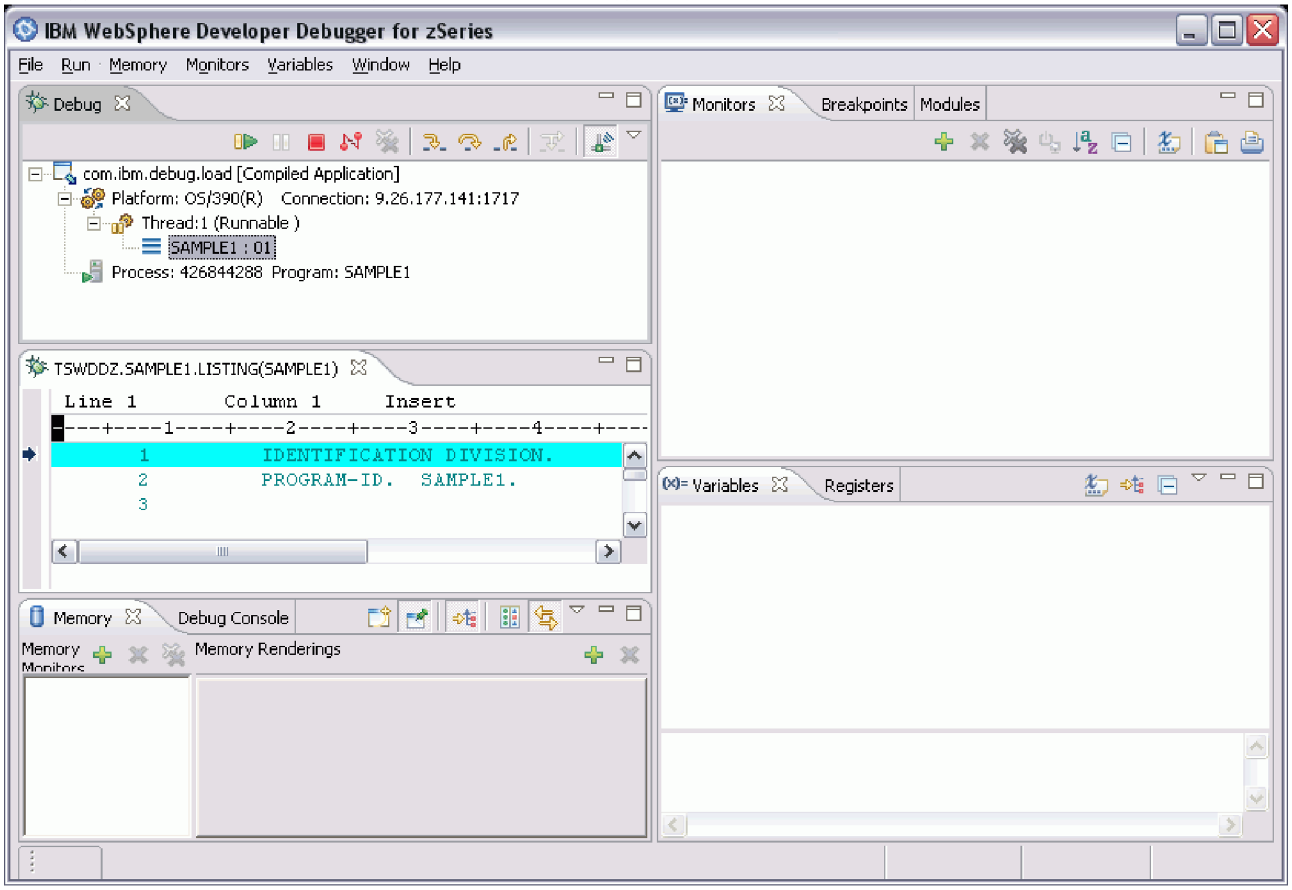
Part 1: Preparing to debug

This tutorial assumes that you have a basic knowledge of MVS, creating an MVS dataset, adding/editing dataset members, MVS JCL, and submitting an MVS job.

Sample code for the application that this tutorial describes is provided with WebSphere® Developer Debugger for zSeries®. You can compile this sample code on the zSeries machine and then create a JCL for it. Ensure that the JCL references the host name of the client machine that you have installed WebSphere Developer Debugger for zSeries on and the daemon port number that you have set in the WebSphere Developer Debugger for zSeries user interface. Submit the JCL on the zSeries machine to begin the debug session on the client machine.

© Copyright IBM Corp. 2005

When you launch the debugger, its welcome experience will appear. Close the welcome panel to see the debugger user interface. After you launch the debug session, the user interface will look like this:



```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)   Connection: 9.26.177.141:1717
    Thread:1 (Runnable )
      SAMPLE1 : 01
  Process: 426844288  Program: SAMPLE1
```

TSWDDZ.SAMPLE1.LISTING(SAMPLE1)

```
 Line 1         Column 1      Insert
----+----1----+----2----+----3----+----4----+----
                IDENTIFICATION DIVISION.
                PROGRAM-ID.  SAMPLE1.
```

Note: If you receive a message indicating that the environment is not yet fully initialized, click **OK** to close it.

Part 2: Get accustomed to the user interface

The WebSphere Developer Debugger for zSeries user interface contains a variety of views that are tailored to help you with typical debugging tasks. These views include:

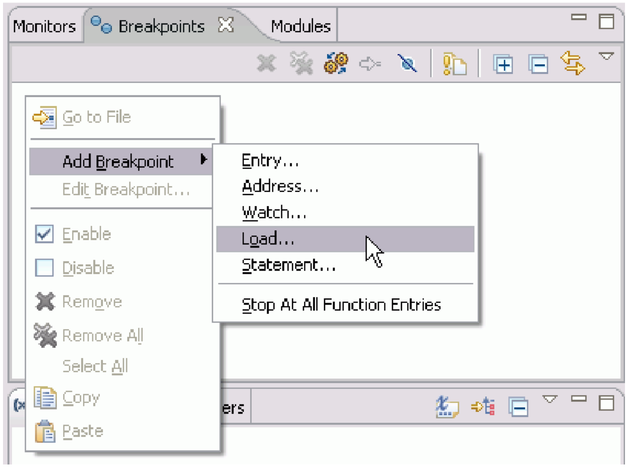
- The Debug view, which allows you to manage the debug session. The Debug view is where you issue run, step, and terminate commands.
- The Breakpoints view, which contains a list of breakpoints that you have set. You can also set breakpoints in this view.
- The Monitors view, which displays a list of variables and expressions that you have added for monitoring. The Variables view displays all current variables and expressions. If you are interested in working with particular variables, the Monitors view is a convenient place to do so.
- The Debug Console, which displays output from Debug Tool for z/OS®. You can also enter commands to Debug Tool for z/OS from the Debug Console.
- The editor, which displays source for your program at the current execution point. You can set statement breakpoints from the editor, as well as issuing run to location and jump to location actions.

As you are following this tutorial, you are encouraged to look at all of the views in the debugger user interface so you can get accustomed to the user interface. To see a view that is hidden. select its tab. In addition, select the main menu bar items during the tutorial. There you will see a variety of debug actions, as well as options to open debug preferences, the debugger welcome page, and the help.

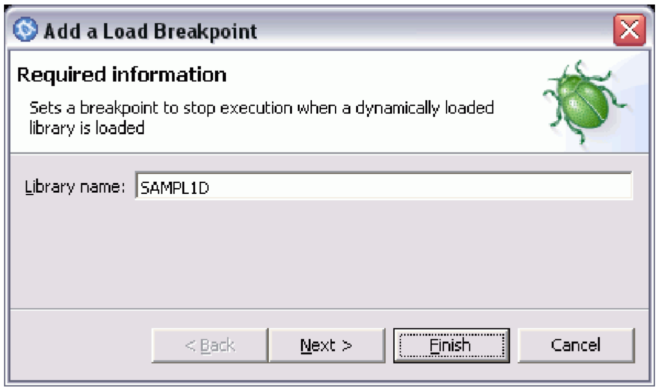
Part 3: Debugging the application

This section will walk you through some basic debug functions. You will set breakpoints, issue step commands, work with source, and examine variables (values for variables may not appear exactly as illustrated).

1. Begin by going to the Breakpoints view. In the view, add a load breakpoint for library SAMPL1D by right-clicking in the view and selecting **Add Breakpoint > Load** from the pop-up menu:



2. Once in the **Add a Load Breakpoint** wizard, enter SAMPL1D in the **Library name** field:



Required information

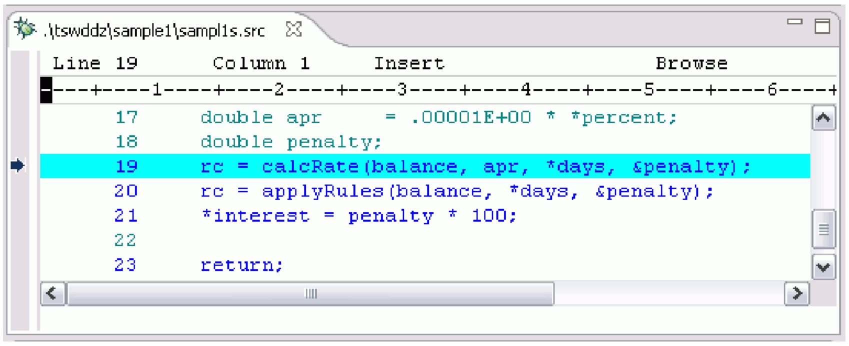
Sets a breakpoint to stop execution when a dynamically loaded library is loaded

Library name: SAMPL1D

Click **Finish** to set the breakpoint.

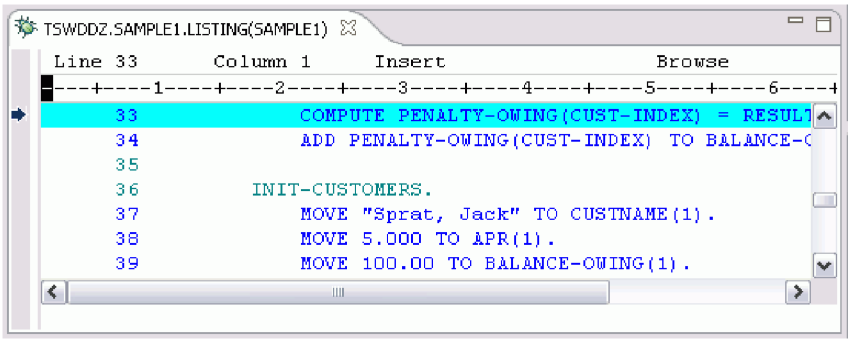
3. Go the Debug view and click **Resume** ( ). The debugger will stop at the load breakpoint that you set.

4. Now look at the editor and you will see that the program is stopped at line 19 in the C subroutine:



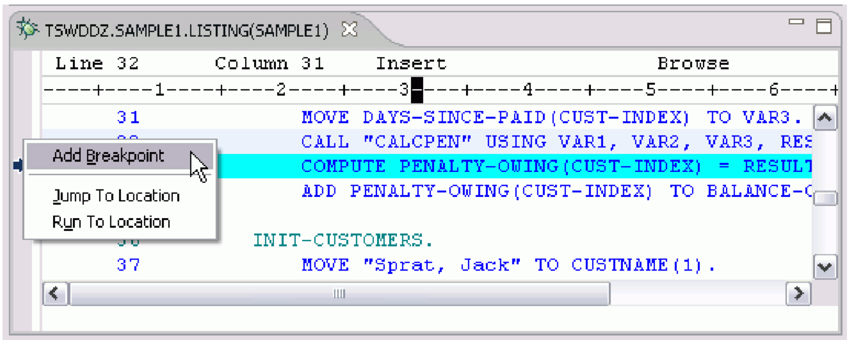
```
 Line 19       Column 1      Insert                  Browse
 ---+----1----+----2----+----3----+----4----+----5----+----6----+
      17      double apr      = .00001E+00 * *percent;
      18      double penalty;
      19      rc = calcRate(balance, apr, *days, &penalty);
      20      rc = applyRules(balance, *days, &penalty);
      21      *interest = penalty * 100;
      22
      23      return;
```

5. Click **Step Return** ( ) in the Debug view and then view the results in the editor:



```
TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
 Line 33       Column 1      Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
     33              COMPUTE PENALTY-OWING(CUST-INDEX) = RESULT
     34              ADD PENALTY-OWING(CUST-INDEX) TO BALANCE-C
     35
     36          INIT-CUSTOMERS.
     37              MOVE "Sprat, Jack" TO CUSTNAME(1).
     38              MOVE 5.000 TO APR(1).
     39              MOVE 100.00 TO BALANCE-OWING(1).
```

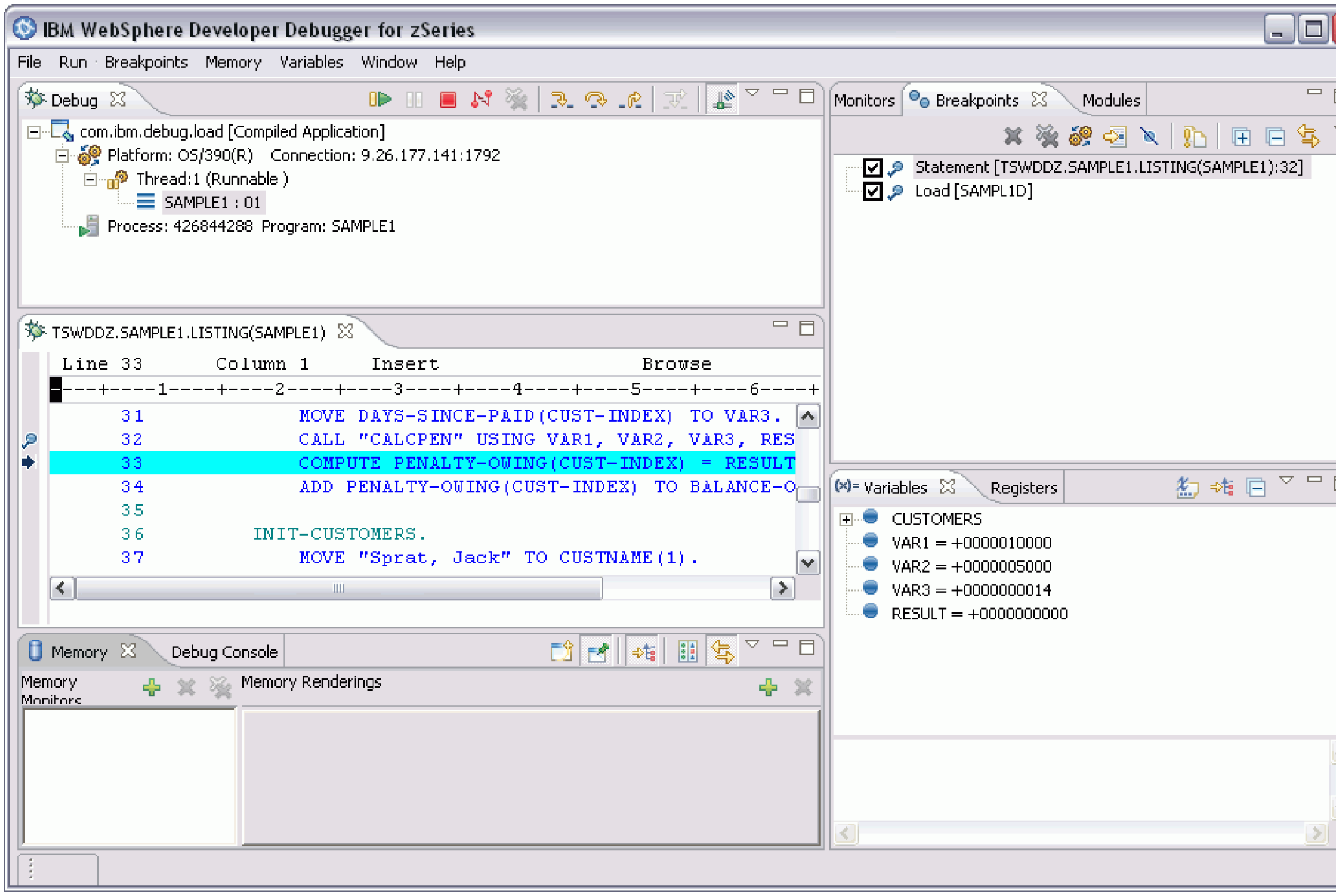
6. Set a breakpoint on line 32. To do this, right-click on the editor marker bar to the left of line 32 and select **Add Breakpoint** from the pop-up menu:



```
TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
 Line 32       Column 31     Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
     31              MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3.
                     CALL "CALCPEN" USING VAR1, VAR2, VAR3, RES
                     COMPUTE PENALTY-OWING(CUST-INDEX) = RESULT
                     ADD PENALTY-OWING(CUST-INDEX) TO BALANCE-C

                 INIT-CUSTOMERS.
     37              MOVE "Sprat, Jack" TO CUSTNAME(1).
```

7. The breakpoint indicator appears in the Breakpoints view and to the left of the line in the editor. The check mark on the breakpoint indicator signifies that the breakpoint is installed. The filled breakpoint indicator (and check mark in the box to the left of the breakpoint in the Breakpoints view) indicates that the breakpoint is enabled.


IBM WebSphere Developer Debugger for zSeries
File Run Breakpoints Memory Variables Window Help
Debug
Monitors Breakpoints Modules
TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
Variables Registers
Memory Debug Console
Memory Monitors
Memory Renderings

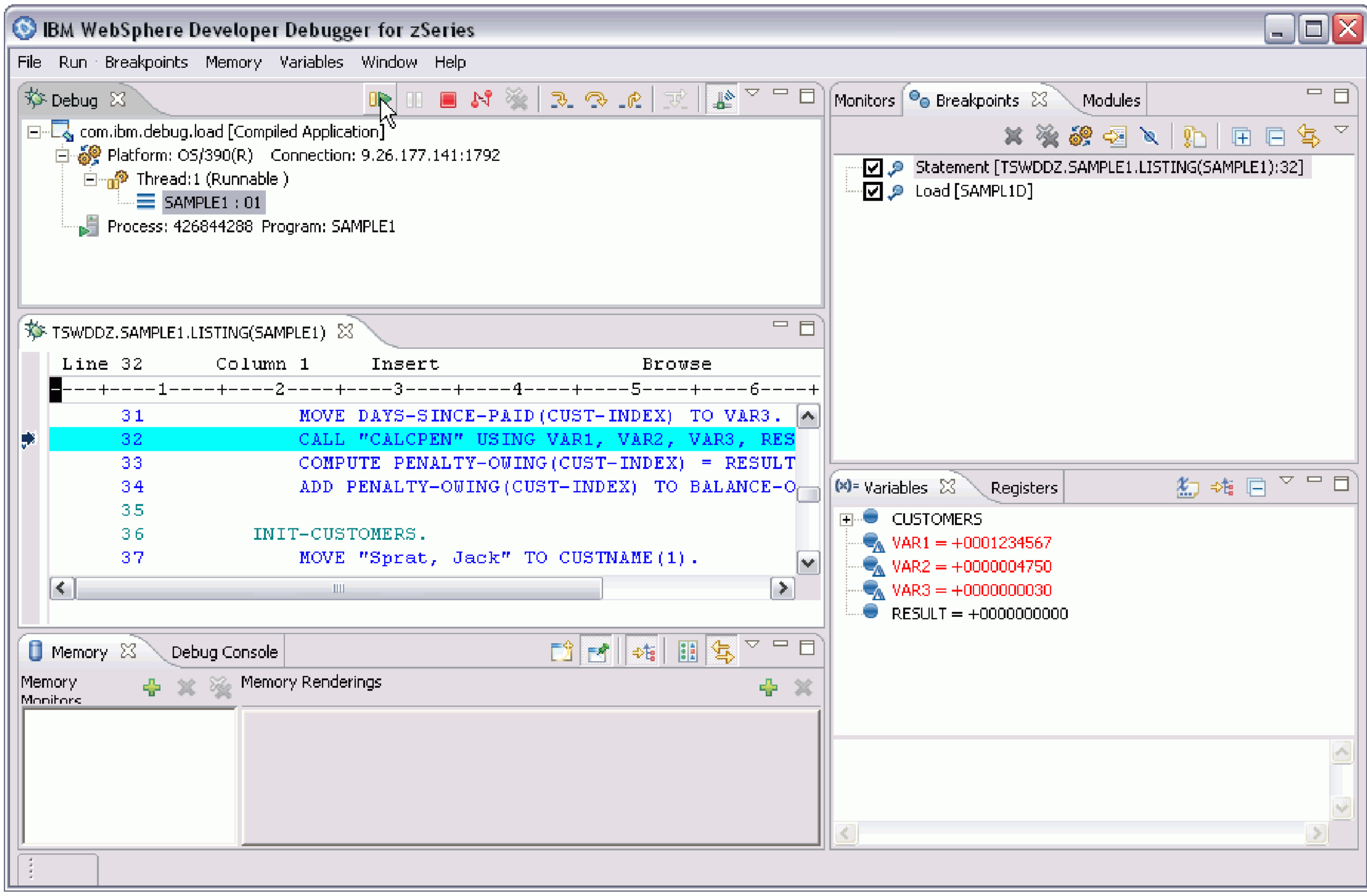
```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)   Connection: 9.26.177.141:1792
    Thread:1 (Runnable )
      SAMPLE1 : 01
  Process: 426844288  Program: SAMPLE1
```

```
Statement [TSWDDZ.SAMPLE1.LISTING(SAMPLE1):32]
Load [SAMPL1D]
```

```
 Line 33       Column 1      Insert              Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
      31             MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3.
      32             CALL "CALCPEN" USING VAR1, VAR2, VAR3, RES
      33             COMPUTE PENALTY-OWING(CUST-INDEX) = RESULT
      34             ADD PENALTY-OWING(CUST-INDEX) TO BALANCE-O
      35
      36         INIT-CUSTOMERS.
      37             MOVE "Sprat, Jack" TO CUSTNAME(1).
```

```
CUSTOMERS
VAR1 = +0000010000
VAR2 = +0000005000
VAR3 = +0000000014
RESULT = +0000000000
```

8. Click **Resume** ( ) and you will see that program execution stops and the editor displays line 32:


IBM WebSphere Developer Debugger for zSeries
File Run Breakpoints Memory Variables Window Help
Debug
Monitors
Breakpoints
Modules
TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
Variables
Registers
Memory
Debug Console
Memory Monitors
Memory Renderings

```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)   Connection: 9.26.177.141:1792
    Thread:1 (Runnable )
      SAMPLE1 : 01
  Process: 426844288  Program: SAMPLE1
```

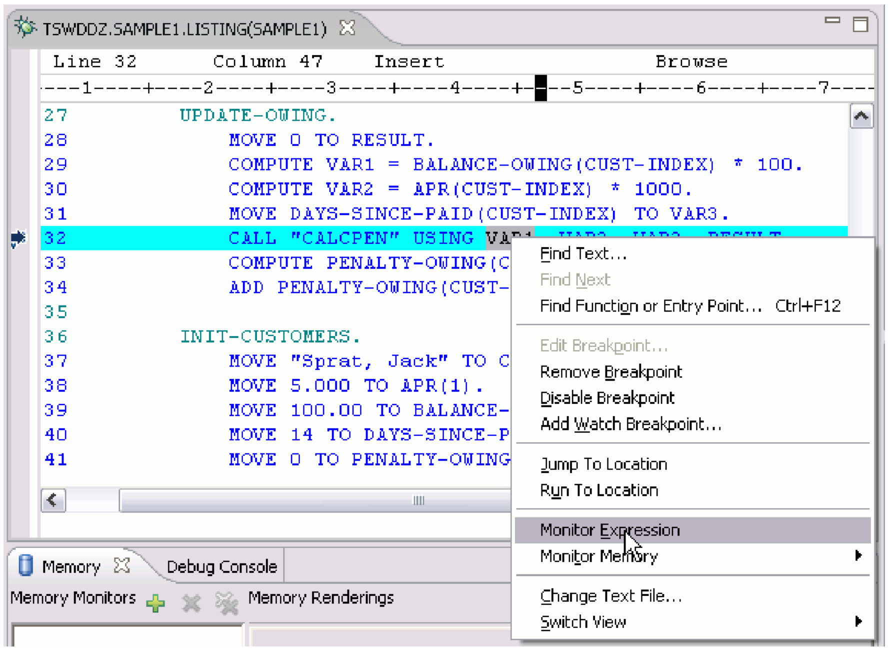
```
Statement [TSWDDZ.SAMPLE1.LISTING(SAMPLE1):32]
Load [SAMPL1D]
```

```
 Line 32       Column 1       Insert                    Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
                  MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3.
                  CALL "CALCPEN" USING VAR1, VAR2, VAR3, RES
                  COMPUTE PENALTY-OWING(CUST-INDEX) = RESULT
                  ADD PENALTY-OWING(CUST-INDEX) TO BALANCE-O

              INIT-CUSTOMERS.
                  MOVE "Sprat, Jack" TO CUSTNAME(1).
```

```
CUSTOMERS
VAR1 = +0001234567
VAR2 = +0000004750
VAR3 = +0000000030
RESULT = +0000000000
```

9. Add variables VAR1, VAR2, VAR3, and RESULT to the Monitors view. To do this, highlight the variable in the editor, right-click, and select **Monitor Expression** from the pop-up menu:

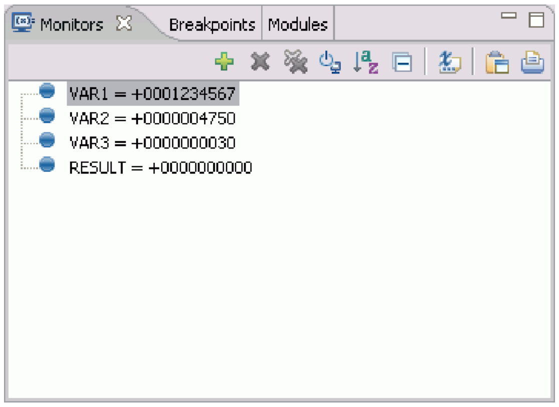


```
 Line 32       Column 47      Insert                    Browse
---1----+----2----+----3----+----4----+----5----+----6----+----7----
          UPDATE-OWING.
              MOVE 0 TO RESULT.
              COMPUTE VAR1 = BALANCE-OWING(CUST-INDEX) * 100.
              COMPUTE VAR2 = APR(CUST-INDEX) * 1000.
              MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3.
              CALL "CALCPEN" USING VAR1
              COMPUTE PENALTY-OWING(C
              ADD PENALTY-OWING(CUST-

          INIT-CUSTOMERS.
              MOVE "Sprat, Jack" TO C
              MOVE 5.000 TO APR(1).
              MOVE 100.00 TO BALANCE-
              MOVE 14 TO DAYS-SINCE-P
              MOVE 0 TO PENALTY-OWING
```

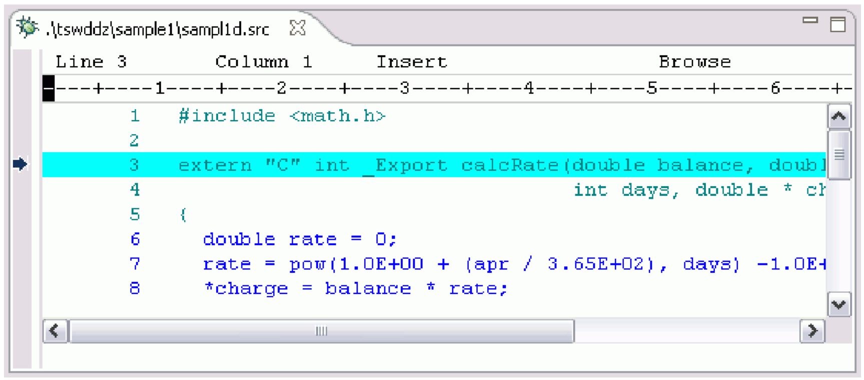
Repeat this for the all of the variables (VAR1, VAR2, VAR3, and RESULT).

10. The variables will display in the Monitors view:



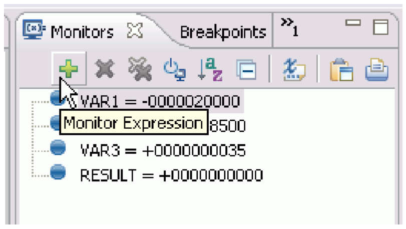
```
VAR1 = +0001234567
VAR2 = +0000004750
VAR3 = +0000000030
RESULT = +0000000000
```

11. Click **Step Into** ( ) in the Debug view until program execution stops at line 3 (`calcRate`) in the editor.
12. Look at the editor to see that program execution has stopped at the `calcRate` entry point in the DLL:



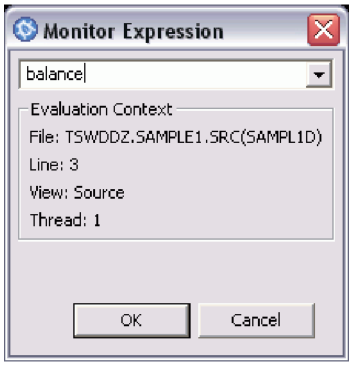
```
 Line 3          Column 1        Insert                       Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+-
        1   #include <math.h>
        2
        3   extern "C" int _Export calcRate(double balance, doubl
        4                                    int days, double * ch
        5   {
        6     double rate = 0;
        7     rate = pow(1.0E+00 + (apr / 3.65E+02), days) -1.0E+
        8     *charge = balance * rate;
```

13. Next, you will add monitors for the `balance`, `apr`, `days`, and `*charge` variables. This time, you will add the monitor directly from the Monitors view. Click the **Monitor Expression** icon at the top of the view:



```
VAR1 = -0000020000
          8500
VAR3 = +0000000035
RESULT = +0000000000
```

This will open the **Monitor Expression** dialog box. In the dialog box, enter the variable in the field and click **OK**:

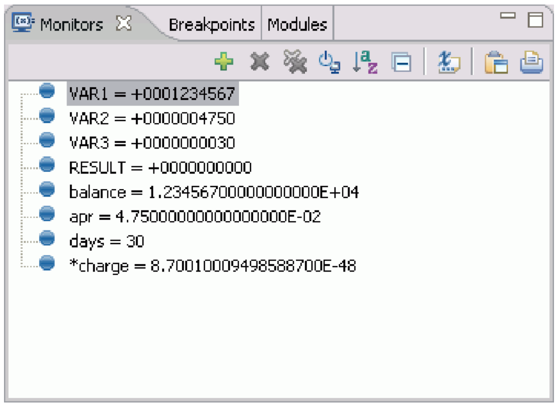


balance

Evaluation Context
File: TSWDDZ.SAMPLE1.SRC(SAMPL1D)
Line: 3
View: Source
Thread: 1

Repeat this for all of the variables (`balance`, `apr`, `days`, and `*charge`).

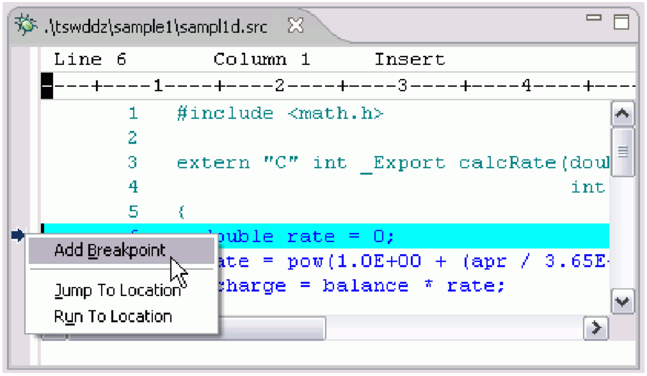
14. After you have added the variables to the Monitors view, they will be displayed in the view in a list:



```
VAR1 = +0001234567
VAR2 = +0000004750
VAR3 = +0000000030
RESULT = +0000000000
balance = 1.23456700000000000E+04
apr = 4.75000000000000000E-02
days = 30
*charge = 8.70010009498588700E-48
```

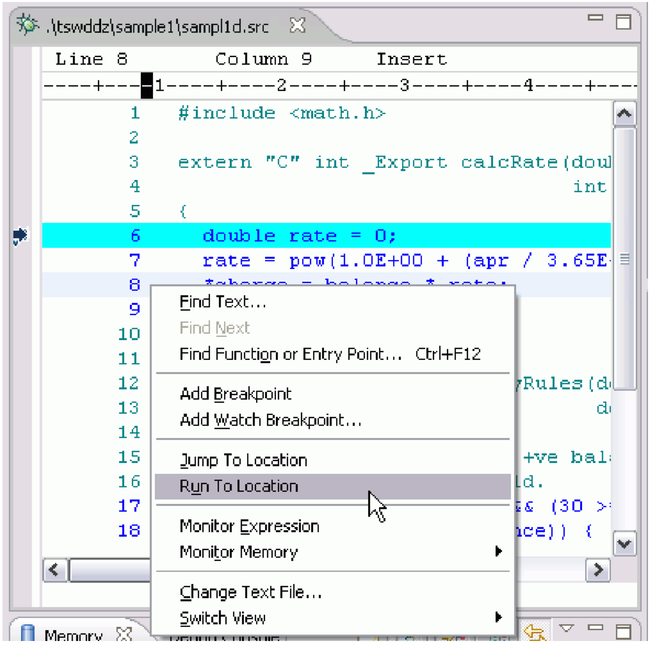
Compare the values of corresponding variables from the C subroutine to those in the C++ DLL and you will see that they match.

15. Click **Step Into** ( ) in the Debug view.
16. Add the `rate` variable to the Monitors view as you did in previous steps. Then, set a breakpoint on the line 6:



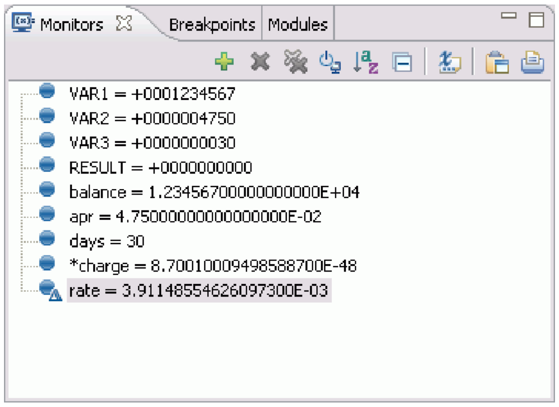
```
 Line 6        Column 1      Insert
----+----1----+----2----+----3----+----4----+---
         1   #include <math.h>
         2
         3   extern "C" int _Export calcRate(dou
         4                                  int
         5   {
               ouble rate = 0;
              ate = pow(1.0E+00 + (apr / 3.65E
              harge = balance * rate;
```

17. Next, you will run to line 8 and then examine the value of `rate`. To do this, select line 8 and then right-click on line 8 in the editor and choose **Run To Location**:



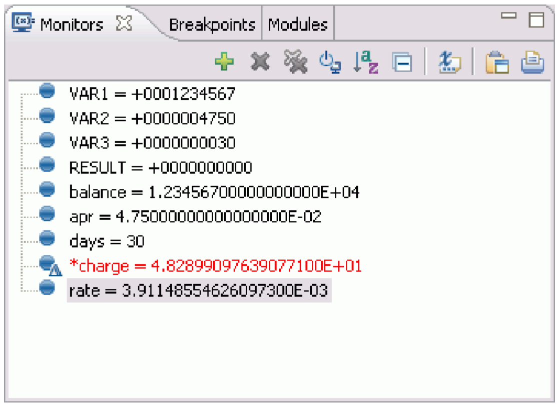
```
 Line 8        Column 9      Insert
----+----1----+----2----+----3----+----4----+---
         1   #include <math.h>
         2
         3   extern "C" int _Export calcRate(dou
         4                                  int
         5   {
         6     double rate = 0;
         7     rate = pow(1.0E+00 + (apr / 3.65E
         8     *charge = balance * rate;
         9
        10
        11
        12                              yRules(d
        13                                     d
        14
        15                              +ve bal
        16                              ld.
        17                              && (30 >
        18                              nce)) {
```

18. Examine the value of `rate` in the Monitors view. You will see that its value has changed. To indicate the change, a delta triangle will appear on the variable indicator:



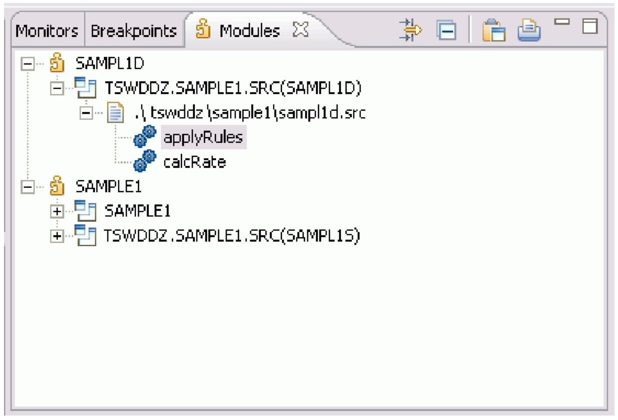
```
VAR1 = +0001234567
VAR2 = +0000004750
VAR3 = +0000000030
RESULT = +0000000000
balance = 1.2345670000000000E+04
apr = 4.7500000000000000E-02
days = 30
*charge = 8.7001000949858870OE-48
rate = 3.9114855462609730OE-03
```

19. Click **Step Into** ( ) in the Debug view.
20. Now the value of `*charge` will indicate a change:



```
VAR1 = +0001234567
VAR2 = +0000004750
VAR3 = +0000000030
RESULT = +0000000000
balance = 1.2345670000000000E+04
apr = 4.7500000000000000E-02
days = 30
*charge = 4.8289909763907710OE+01
rate = 3.9114855462609730OE-03
```

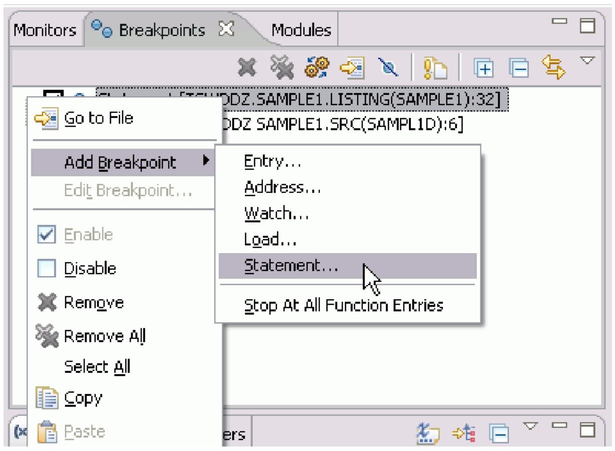
21. Next, you will add a statement breakpoint on the first executable line (line 17) of `applyRules` function in the DLL. However, this time you will make the breakpoint conditional - you will set it to trigger on the third time the statement is executed. First, let's have a look at the `applyRules` function. Go to the Modules view and expand the `SAMPL1D` tree node and its sub-elements until you see the `applyRules` function. Double-click `applyRules`:



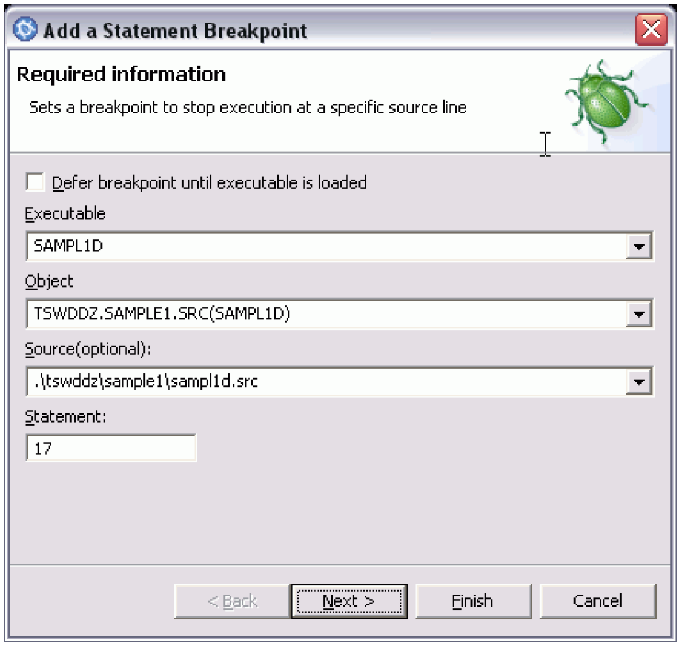
Monitors
Breakpoints
Modules
SAMPL1D
TSWDDZ.SAMPLE1.SRC(SAMPL1D)
.\tswddz\sample1\sampl1d.src
applyRules
calcRate
SAMPLE1
SAMPLE1
TSWDDZ.SAMPLE1.SRC(SAMPL1S)

When you double-click the entry in the Modules view, the editor displays the function with the entry highlighted. Single-click on line 17 to select the line.

22. Now, we will add the conditional statement breakpoint. Right-click in Breakpoints view and select **Add Breakpoint > Statement** from the pop-up menu:



23. In the resulting **Add a Statement Breakpoint** dialog box, you will see that the **Executable**, **Object**, and **Source (optional)** fields are pre-filled with the appropriate information. Make sure that the **Statement** field entry is 17 and then click **Next**:


Add a Statement Breakpoint

Required information

Sets a breakpoint to stop execution at a specific source line

Defer breakpoint until executable is loaded

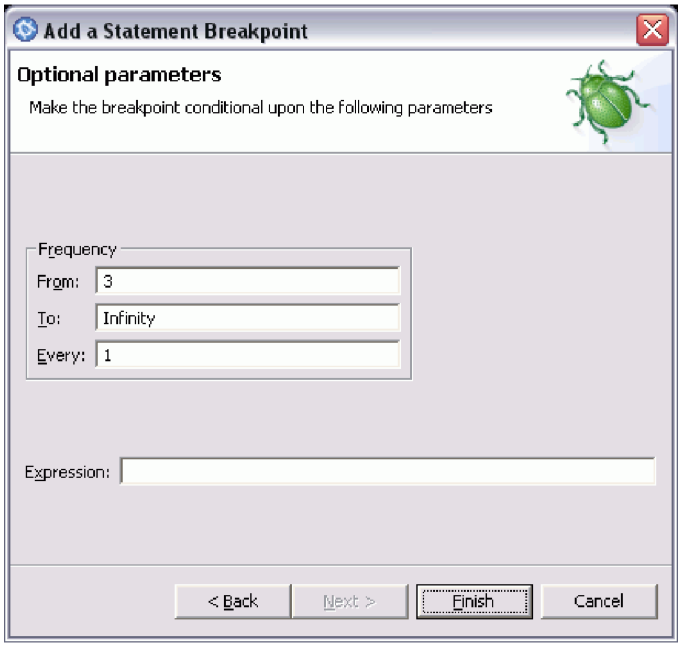
Executable: SAMPL1D

Object: TSWDDZ.SAMPLE1.SRC(SAMPL1D)

Source(optional): .\tswddz\sample1\sampl1d.src

Statement: 17

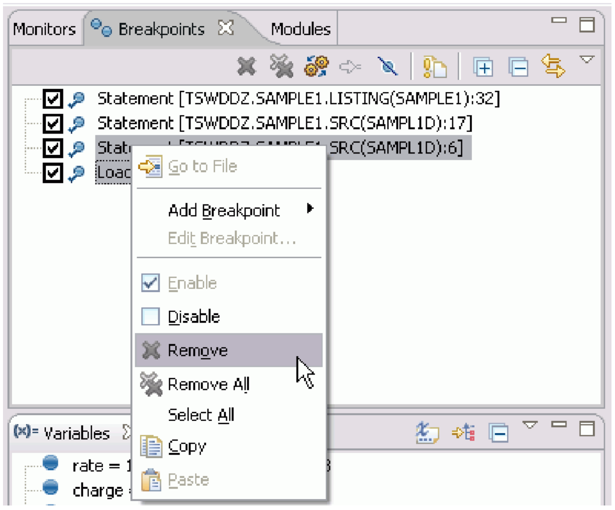
24. Now you are in the Optional parameters page of the **Add a Statement Breakpoint** wizard. This page allows you to set conditions for the breakpoint. We are going to set the breakpoint to trigger on the third time that the statement is executed (we want to see what happens in the fourth execution and we have already gone through this statement once). To do this, change the value of the **From** field to 3 and click **Finish**:



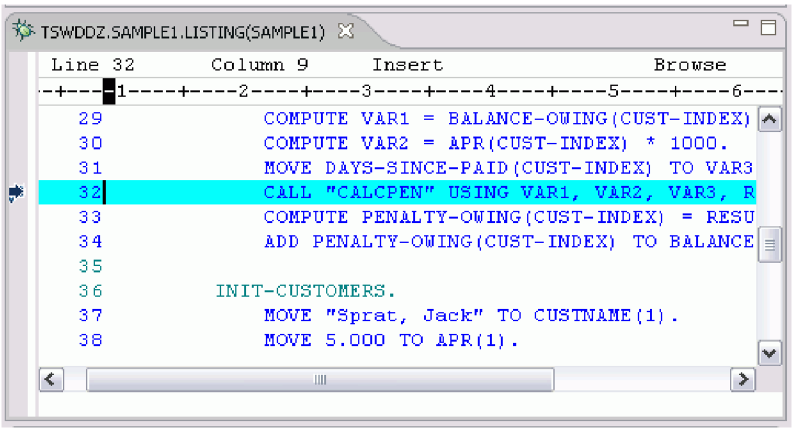
When entering optional breakpoint parameters, the fields in the wizard page are:

- **From**: Enter the first breakpoint encounter you want the debugger to stop on. For example, if you want the debugger to skip over the breakpoint the first five times it is encountered, enter "6".
- **To**: Enter the last breakpoint encounter you want the debugger to stop on. For example, if you want it to start ignoring the breakpoint after the 20th encounter, enter "20". To have it always stop on the breakpoint, enter "Infinity".
- **Every**: Enter the frequency with which you want the debugger to stop on this breakpoint. For example, if you want it to stop on only one out of every four it encounters, enter "4".
- **Expression**: You can enter an expression into this field. The execution of the program stops at the breakpoint only if the condition specified in this field tests true (any non-zero value is considered true).

25. Since we are done with some of the breakpoints that we have been working with, we will delete them. Select all breakpoints except the statement breakpoints for lines 17 and 32. To select multiple breakpoints, use the keyboard Ctrl or Shift keys. Right-click the selection and choose **Remove** from the pop-up menu:



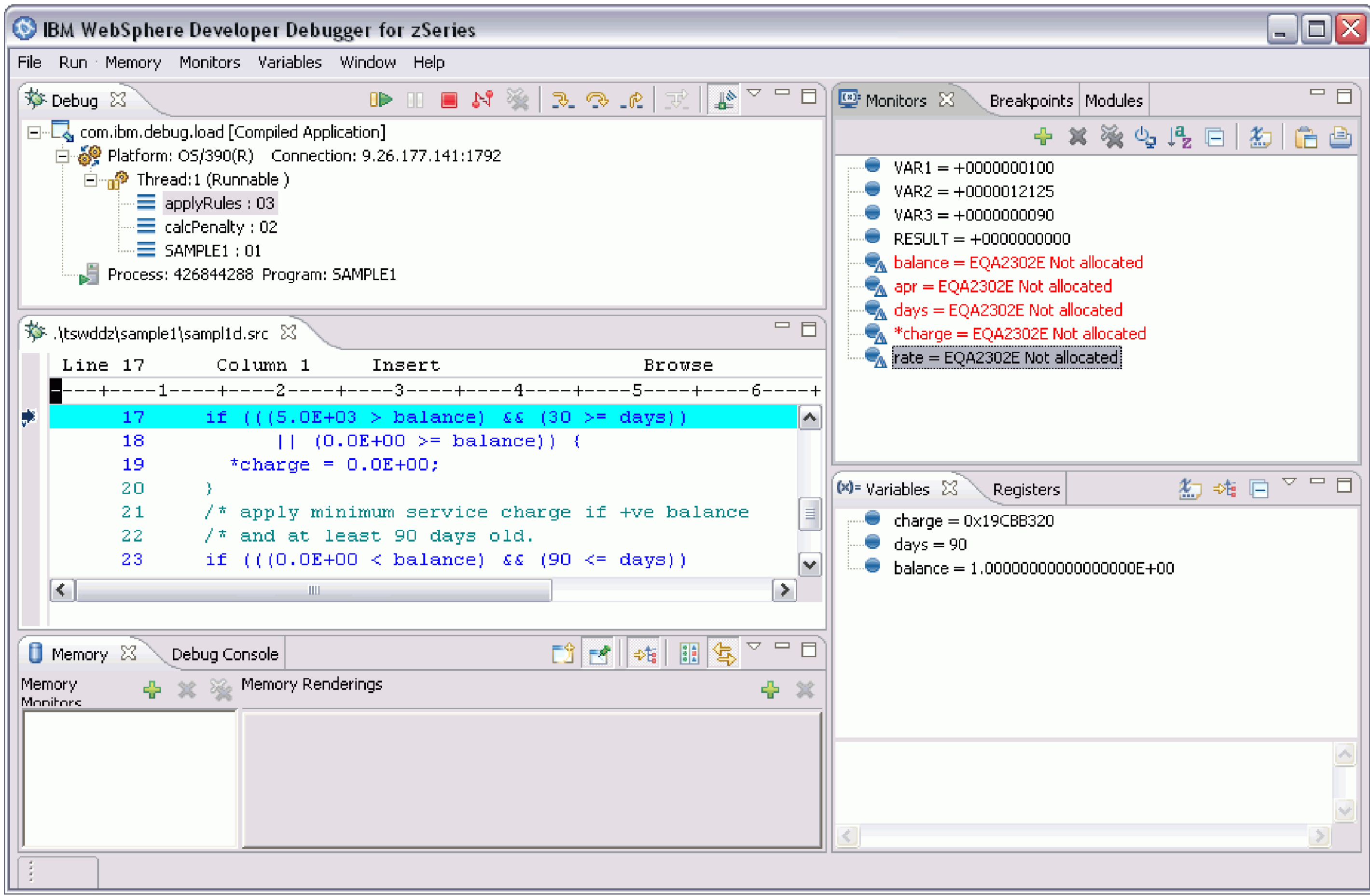
26. Click **Resume** ( ) and you will see that program execution stops and the editor displays line 32:



TSWDDZ.SAMPLE1.LISTING(SAMPLE1)

```
Line 32         Column 9        Insert                  Browse
-+----1----+----2----+----3----+----4----+----5----+----6---
 29              COMPUTE VAR1 = BALANCE-OWING(CUST-INDEX)
 30              COMPUTE VAR2 = APR(CUST-INDEX) * 1000.
 31              MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3
 32              CALL "CALCPEN" USING VAR1, VAR2, VAR3, R
 33              COMPUTE PENALTY-OWING(CUST-INDEX) = RESU
 34              ADD PENALTY-OWING(CUST-INDEX) TO BALANCE
 35
 36          INIT-CUSTOMERS.
 37              MOVE "Sprat, Jack" TO CUSTNAME(1).
 38              MOVE 5.000 TO APR(1).
```

27. Click **Step Over** ( ) in the Debug view and then click **Resume** ( ).
28. Execution halts at the conditional breakpoint that you set earlier. Have a look at the Monitors view and you will see that the values of the `balance`, `apr`, `days`, `*charge`, and `rate` variables are *Not allocated*. This is correct as those variables are being monitored in the scope of the other function in this DLL.



IBM WebSphere Developer Debugger for zSeries

```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)   Connection: 9.26.177.141:1792
    Thread:1 (Runnable )
      applyRules : 03
      calcPenalty : 02
      SAMPLE1 : 01
  Process: 426844288  Program: SAMPLE1
```

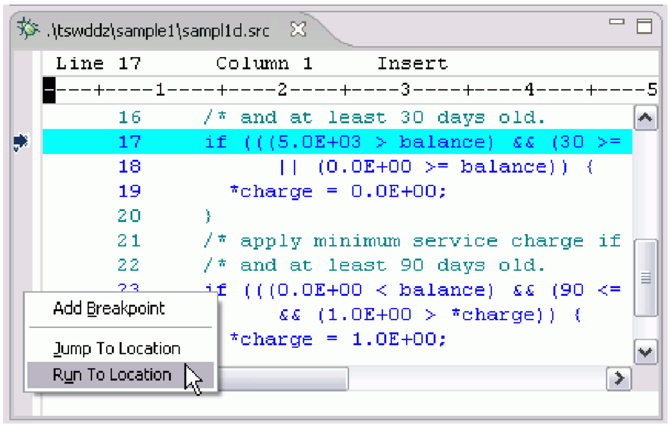
```
VAR1 = +0000000100
VAR2 = +0000012125
VAR3 = +0000000090
RESULT = +0000000000
balance = EQA2302E Not allocated
apr = EQA2302E Not allocated
days = EQA2302E Not allocated
*charge = EQA2302E Not allocated
rate = EQA2302E Not allocated
```

.\tswddz\sample1\sampl1d.src

```
Line 17         Column 1        Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
      17      if (((5.0E+03 > balance) && (30 >= days))
      18            || (0.0E+00 >= balance)) {
      19         *charge = 0.0E+00;
      20      }
      21      /* apply minimum service charge if +ve balance
      22      /* and at least 90 days old.
      23      if (((0.0E+00 < balance) && (90 <= days))
```

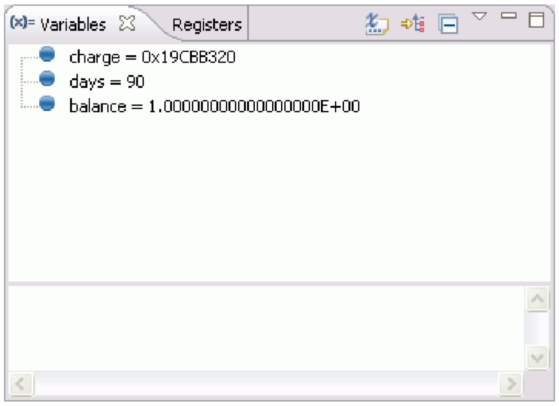
```
charge = 0x19CBB320
days = 90
balance = 1.000000000000000000E+00
```

29. Now, let's run to line 23. In the editor, right-click the marker bar to the left of line 23 and select **Run To Location** from the pop-up menu:



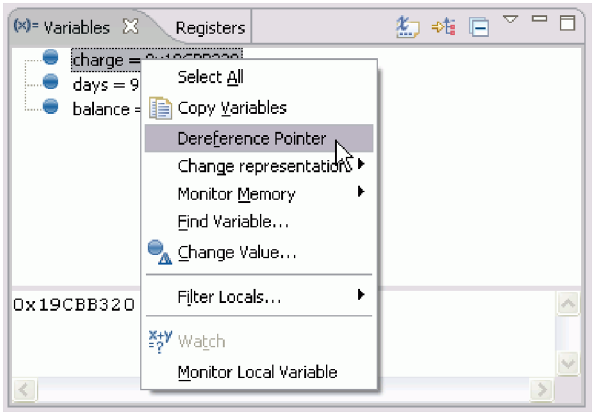
```
.\tswddz\sample1\sampl1d.src
Line 17       Column 1      Insert
-+---+----1----+----2----+----3----+----4----+----5
       /* and at least 30 days old.
       if (((5.0E+03 > balance) && (30 >=
              || (0.0E+00 >= balance)) {
         *charge = 0.0E+00;
       }
       /* apply minimum service charge if
       /* and at least 90 days old.
       if (((0.0E+00 < balance) && (90 <=
              && (1.0E+00 > *charge)) {
         *charge = 1.0E+00;
```

30. Next, go the Variables view and observe the values of the variables after running to line 23 (since charge is an address, its value may be different on your machine than in this screen capture):

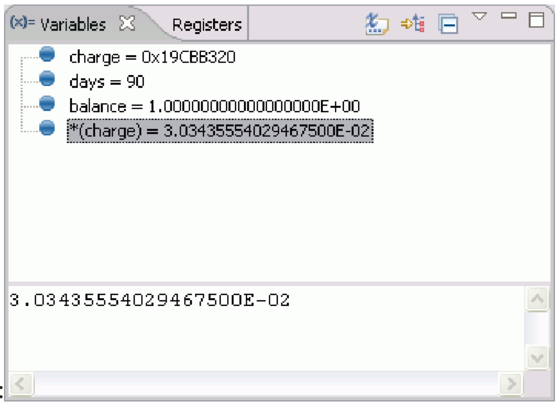


```
charge = 0x19CBB320
days = 90
balance = 1.00000000000000000E+00
```

31. Now you can dereference the `charge` variable. To do this, right-click the variable in the Variables view and select **Dereference Pointer** from the pop-up menu:



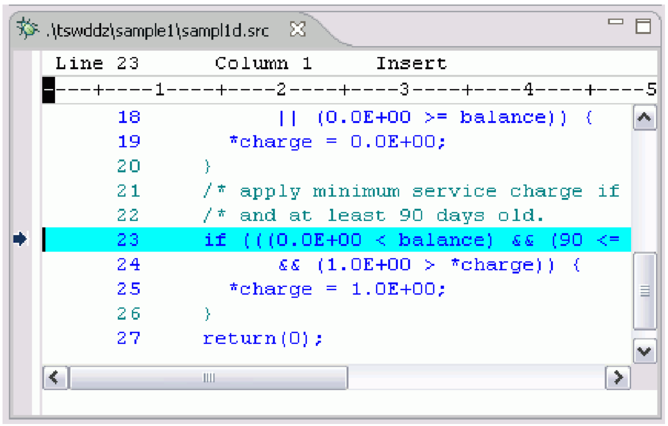
After choosing this action, you will see the dereferenced pointer added to the list in the Variables view:



```
charge = 0x19CBB320
days = 90
balance = 1.000000000000000000E+00
*(charge) = 3.034355540294675000E-02
```

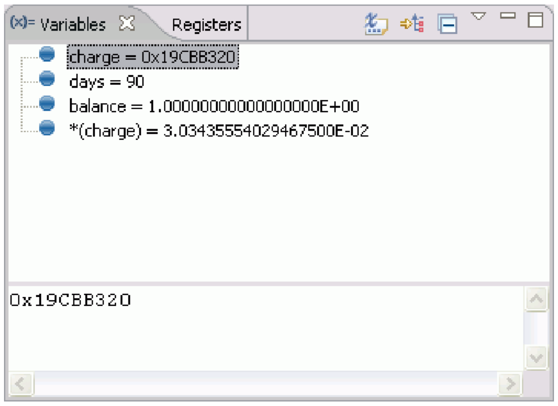
```
3.034355540294675000E-02
```

32. Next, let’s examine the `if` statement at the current execution point:



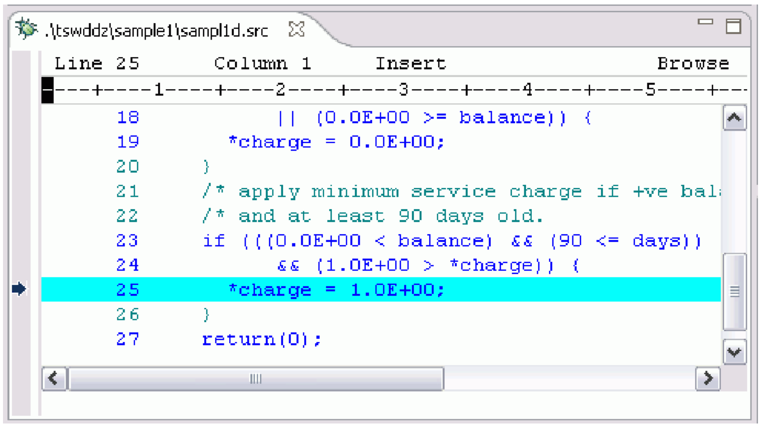
```
Line 23          Column 1       Insert
----+----1----+----2----+----3----+----4----+----5
               || (0.0E+00 >= balance)) {
             *charge = 0.0E+00;
           }
           /* apply minimum service charge if
           /* and at least 90 days old.
           if (((0.0E+00 < balance) && (90 <=
               && (1.0E+00 > *charge)) {
             *charge = 1.0E+00;
           }
           return(0);
```

Click **Step Over** ( ) in the Debug view three times and then have a look at the Variables view and the editor:



```
charge = 0x19CBB320
days = 90
balance = 1.000000000000000000E+00
*(charge) = 3.034355540294675000E-02
```

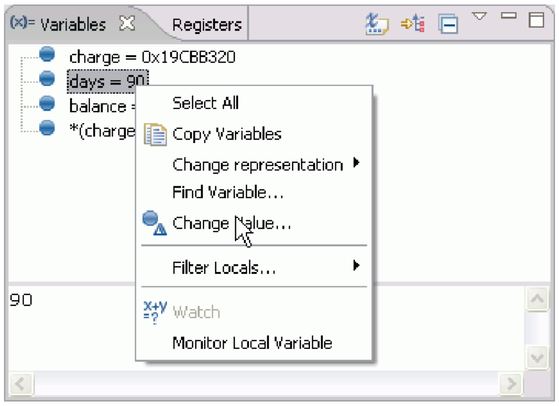
```
0x19CBB320
```



```
Line 25        Column 1       Insert                   Browse
----+----1----+----2----+----3----+----4----+----5----+---
               || (0.0E+00 >= balance)) {
           *charge = 0.0E+00;
         }
         /* apply minimum service charge if +ve bal
         /* and at least 90 days old.
         if (((0.0E+00 < balance) && (90 <= days))
               && (1.0E+00 > *charge)) {
           *charge = 1.0E+00;
         }
         return(0);
```

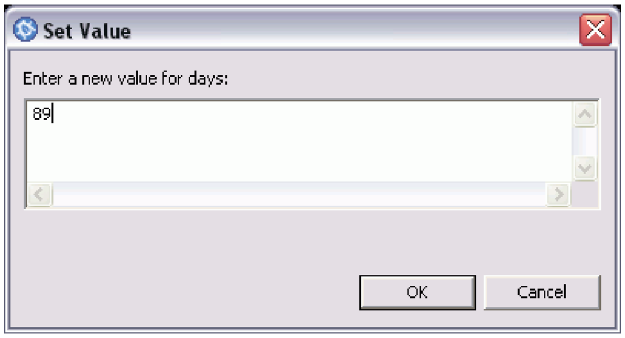
Notice that the three conditions evaluate to true.

33. Now, let's change the value of a variable. In the Variables view, right-click days and select **Change Value** from the pop-up menu:



```
charge = 0x19CBB320
days = 90
balance =
*(charge
90
```

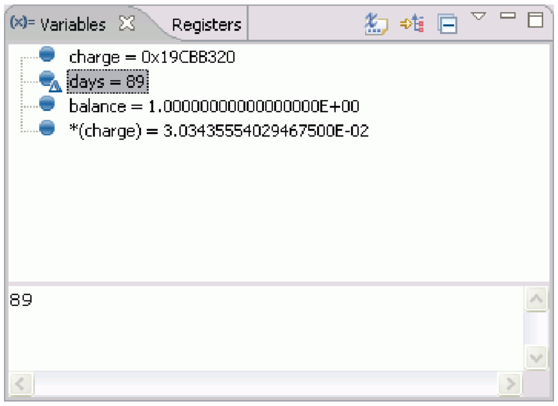
This will open the **Set Value** dialog box. Use this dialog box to change the value of days to 89:



Enter a new value for days:

89

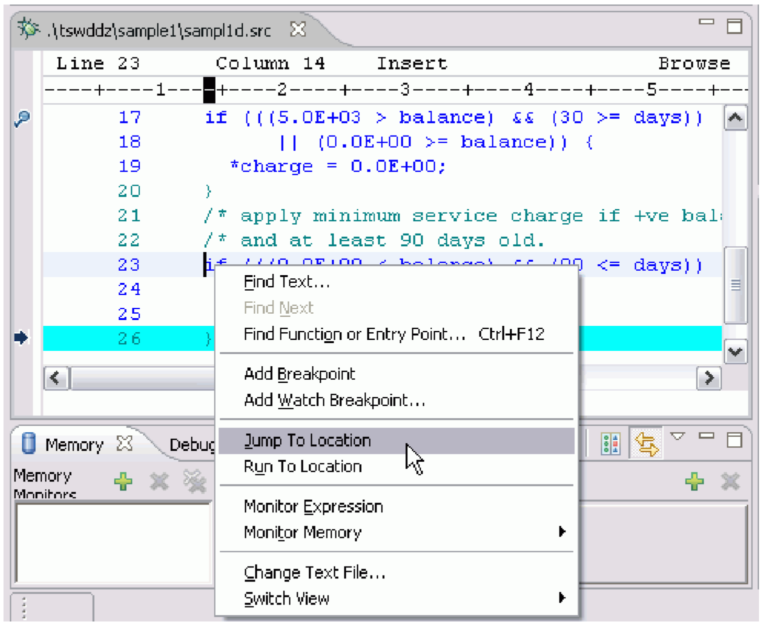
After you click **OK**, you will see that the value of the variable has changed and the variable indicator will have a delta symbol to the next of it:



```
charge = 0x19CBB320
days = 89
balance = 1.000000000000000000E+00
*(charge) = 3.034355540294675OOE-02
```

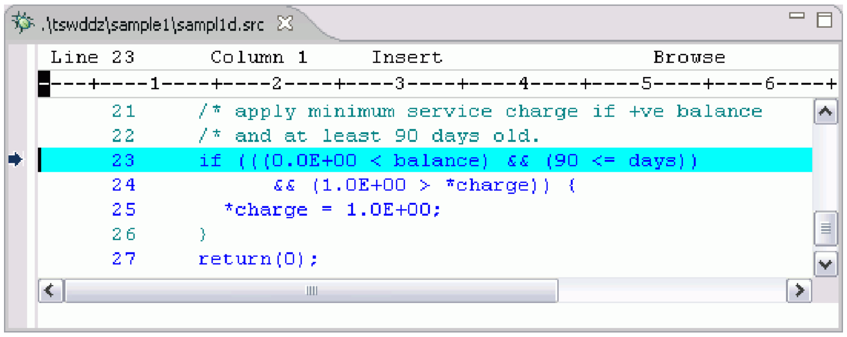
```
89
```

34. Now, let's step out of the `if` statement. To do this, select **Step Over** ( ) or **Step Into** ( ) twice in the Debug view (this will move the execution pointer out of the `if` statement). Then, jump to line 23 and reevaluate the `if` statement. To do this, right-click the marker bar to the left of line 23 in the editor and select **Jump To Location** from the pop-up menu:



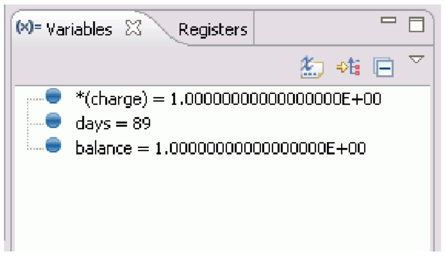
```
 Line 23          Column 14     Insert                          Browse
----+----1----+----2----+----3----+----4----+----5----+---
       17     if (((5.0E+03 > balance) && (30 >= days))
       18           || (0.0E+00 >= balance)) {
       19       *charge = 0.0E+00;
       20     }
       21     /* apply minimum service charge if +ve bal
       22     /* and at least 90 days old.
       23     if (((0.0E+00 < balance) && (90 <= days))
       24
       25
       26     }
```

After issuing the jump to location action, line 23 will be highlighted in the editor (this is the current execution point):

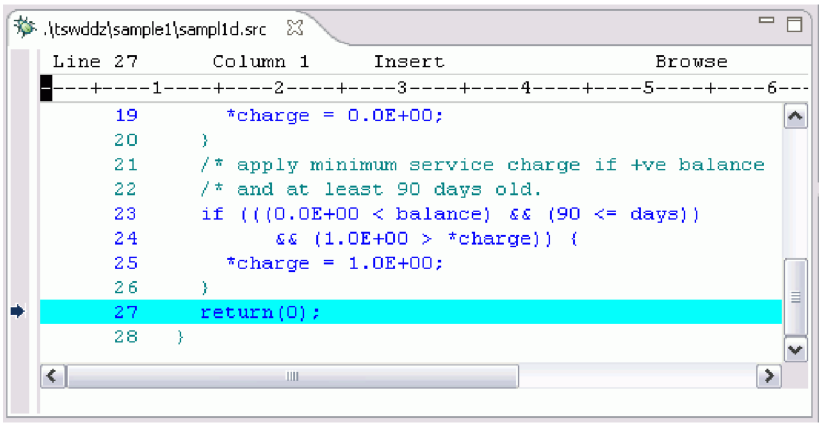


```
 Line 23          Column 1      Insert                          Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+
       21     /* apply minimum service charge if +ve balance
       22     /* and at least 90 days old.
       23     if (((0.0E+00 < balance) && (90 <= days))
       24           && (1.0E+00 > *charge)) {
       25       *charge = 1.0E+00;
       26     }
       27     return(0);
```

35. Examine the `if` statement again. Click **Step Over** ( ) in the Debug view and have a look at the Variables view and the editor:



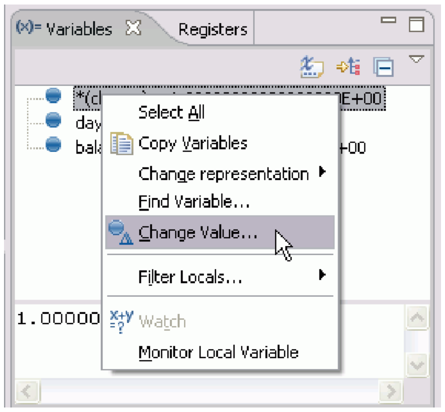
```
*(charge) = 1.000000000000000000E+00
days = 89
balance = 1.000000000000000000E+00
```

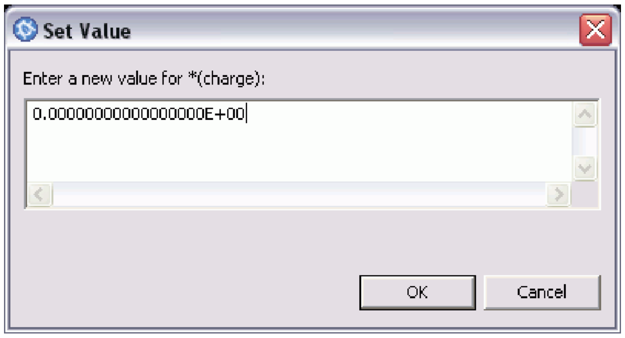


```
Line 27       Column 1       Insert                 Browse
----+----1----+----2----+----3----+----4----+----5----+----6---
            *charge = 0.0E+00;
          }
          /* apply minimum service charge if +ve balance
          /* and at least 90 days old.
          if (((0.0E+00 < balance) && (90 <= days))
                && (1.0E+00 > *charge)) {
            *charge = 1.0E+00;
          }
          return(0);
        }
```

This time, the `if` statement evaluates as false.

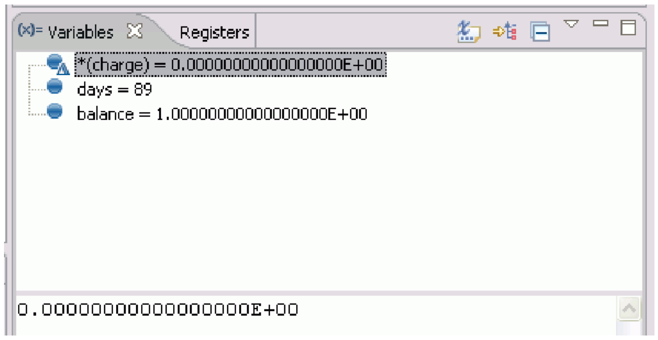
36. Change the value of `*charge` to zero:





Enter a new value for *(charge):

```
0.000000000000000000E+00
```

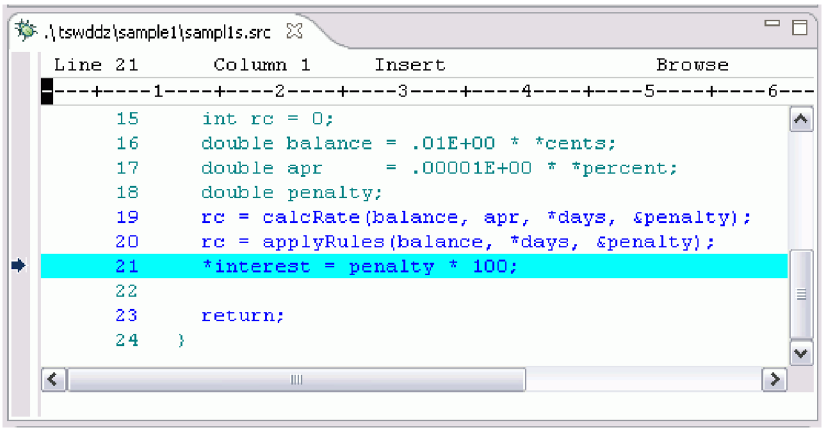


Variables Registers

```
*(charge) = 0.00000000000000000E+00
days = 89
balance = 1.00000000000000000E+00
```

```
0.00000000000000000E+00
```

37. **Step Return** ( ) in the Debug view. This will bring you back to the C subroutine after the call to `applyRules`:

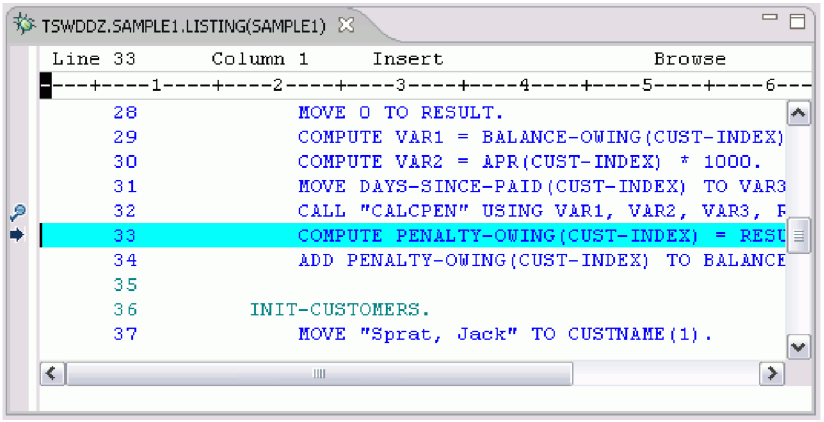


.\tswddz\sample1\sampl1s.src

```
 Line 21       Column 1      Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6---
         int rc = 0;
         double balance = .01E+00 * *cents;
         double apr     = .00001E+00 * *percent;
         double penalty;
         rc = calcRate(balance, apr, *days, &penalty);
         rc = applyRules(balance, *days, &penalty);
         *interest = penalty * 100;

         return;
       }
```

38. Click **Step Return** again. You will then be in the COBOL mainline after the call to `CALCPEN`:

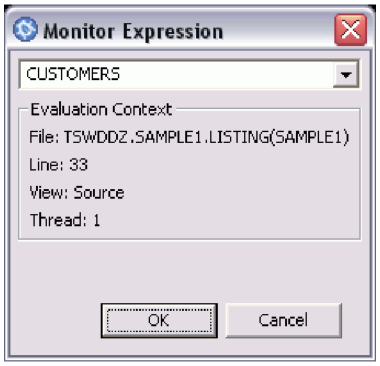


TSWDDZ.SAMPLE1.LISTING(SAMPLE1)

```
 Line 33       Column 1      Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6---
                 MOVE 0 TO RESULT.
                 COMPUTE VAR1 = BALANCE-OWING(CUST-INDEX)
                 COMPUTE VAR2 = APR(CUST-INDEX) * 1000.
                 MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3
                 CALL "CALCPEN" USING VAR1, VAR2, VAR3, R
                 COMPUTE PENALTY-OWING(CUST-INDEX) = RESU
                 ADD PENALTY-OWING(CUST-INDEX) TO BALANCE

             INIT-CUSTOMERS.
                 MOVE "Sprat, Jack" TO CUSTNAME(1).
```

39. Now, let's add `CUSTOMERS` to the Monitors view. To do this, launch the **Monitor Expression** dialog box from the Monitors view:



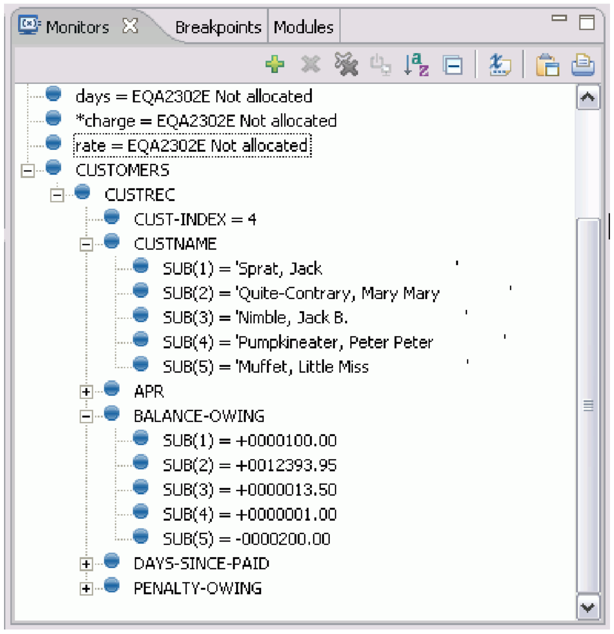
Monitor Expression

CUSTOMERS

Evaluation Context
File: TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
Line: 33
View: Source
Thread: 1

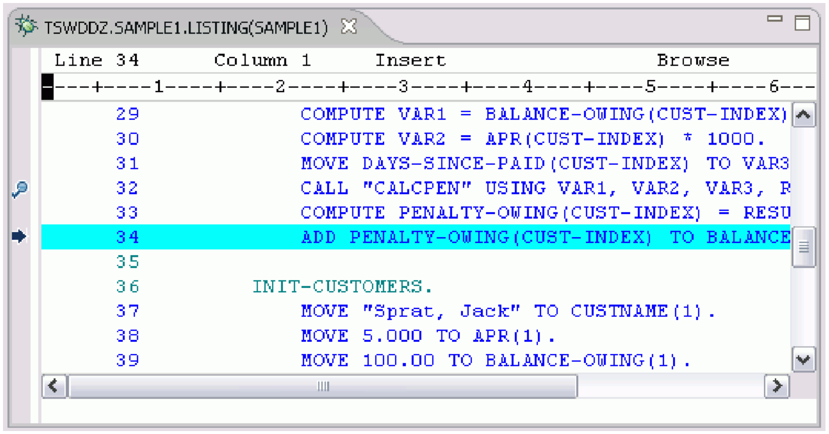
OK Cancel

40. In the Monitors view, expand `CUSTOMERS` and then `CUSTNAME` and `BALANCE-OWING`:



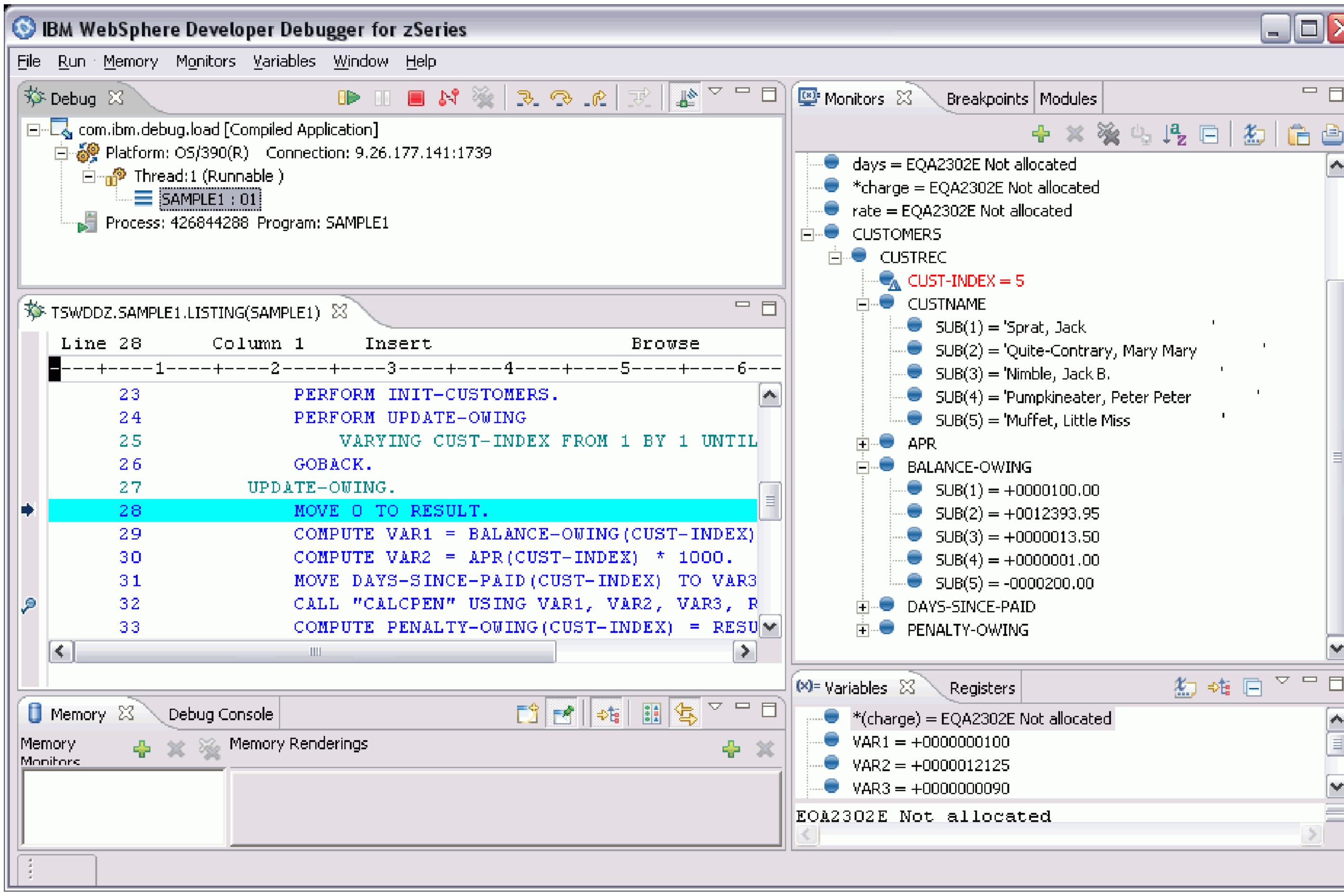
```
days = EQA2302E Not allocated
*charge = EQA2302E Not allocated
rate = EQA2302E Not allocated
CUSTOMERS
  CUSTREC
    CUST-INDEX = 4
    CUSTNAME
      SUB(1) = 'Sprat, Jack                    '
      SUB(2) = 'Quite-Contrary, Mary Mary       '
      SUB(3) = 'Nimble, Jack B.                 '
      SUB(4) = 'Pumpkineater, Peter Peter       '
      SUB(5) = 'Muffet, Little Miss             '
    APR
    BALANCE-OWING
      SUB(1) = +0000100.00
      SUB(2) = +0012393.95
      SUB(3) = +0000013.50
      SUB(4) = +0000001.00
      SUB(5) = -0000200.00
    DAYS-SINCE-PAID
    PENALTY-OWING
```

41. Now, click **Step Over** ( ) in the Debug view and you will be at line 34 in the COBOL source:



```
 Line 34      Column 1      Insert                  Browse
----+----1----+----2----+----3----+----4----+----5----+----6---
      29              COMPUTE VAR1 = BALANCE-OWING(CUST-INDEX)
      30              COMPUTE VAR2 = APR(CUST-INDEX) * 1000.
      31              MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3
      32              CALL "CALCPEN" USING VAR1, VAR2, VAR3, R
      33              COMPUTE PENALTY-OWING(CUST-INDEX) = RESU
      34              ADD PENALTY-OWING(CUST-INDEX) TO BALANCE
      35
      36          INIT-CUSTOMERS.
      37              MOVE "Sprat, Jack" TO CUSTNAME(1).
      38              MOVE 5.000 TO APR(1).
      39              MOVE 100.00 TO BALANCE-OWING(1).
```

42. Click **Step Over** over again and you will notice that `CUSTOMERS.CUSTREC.BALANCE-OWING.SUB(4)` did not change. Its value is still 1.00:



IBM WebSphere Developer Debugger for zSeries
File Run Memory Monitors Variables Window Help
Debug
Monitors
Breakpoints
Modules
TSWDDZ.SAMPLE1.LISTING(SAMPLE1)
Memory
Debug Console
Memory Monitors
Memory Renderings
Variables
Registers

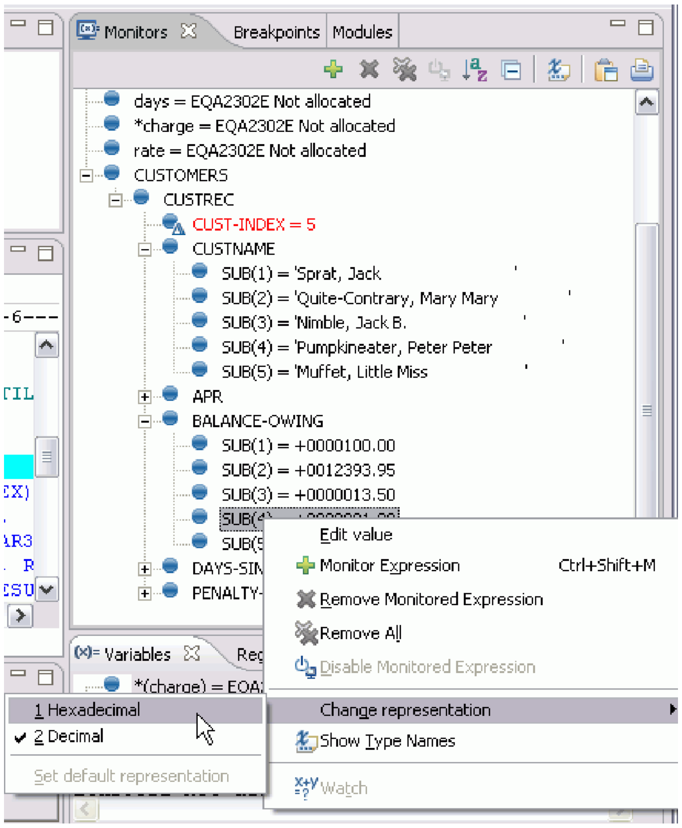
```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)    Connection: 9.26.177.141:1739
    Thread:1 (Runnable )
      SAMPLE1 : 01
  Process: 426844288  Program: SAMPLE1
```

```
 Line 28        Column 1       Insert                        Browse
 ----+----1----+----2----+----3----+----4----+----5----+----6---
                PERFORM INIT-CUSTOMERS.
                PERFORM UPDATE-OWING
                    VARYING CUST-INDEX FROM 1 BY 1 UNTIL
                GOBACK.
            UPDATE-OWING.
                MOVE 0 TO RESULT.
                COMPUTE VAR1 = BALANCE-OWING(CUST-INDEX)
                COMPUTE VAR2 = APR(CUST-INDEX) * 1000.
                MOVE DAYS-SINCE-PAID(CUST-INDEX) TO VAR3
                CALL "CALCPEN" USING VAR1, VAR2, VAR3, R
                COMPUTE PENALTY-OWING(CUST-INDEX) = RESU
```

```
days = EQA2302E Not allocated
*charge = EQA2302E Not allocated
rate = EQA2302E Not allocated
CUSTOMERS
  CUSTREC
    CUST-INDEX = 5
    CUSTNAME
      SUB(1) = 'Sprat, Jack                    '
      SUB(2) = 'Quite-Contrary, Mary Mary      '
      SUB(3) = 'Nimble, Jack B.                '
      SUB(4) = 'Pumpkineater, Peter Peter      '
      SUB(5) = 'Muffet, Little Miss            '
    APR
    BALANCE-OWING
      SUB(1) = +0000100.00
      SUB(2) = +0012393.95
      SUB(3) = +0000013.50
      SUB(4) = +0000001.00
      SUB(5) = -0000200.00
    DAYS-SINCE-PAID
    PENALTY-OWING
```

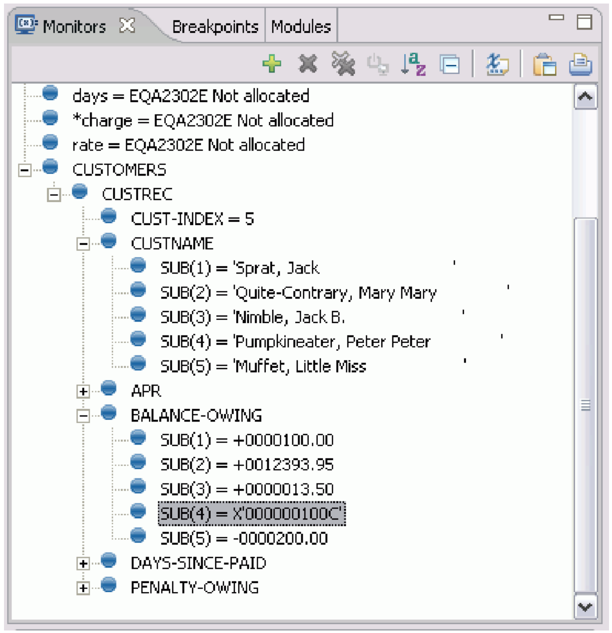
```
*(charge) = EQA2302E Not allocated
VAR1 = +0000000100
VAR2 = +0000012125
VAR3 = +0000000090
EQA2302E Not allocated
```

43. Now, change the representation of `CUSTOMERS.CUSTREC.BALANCE-OWING.SUB(4)` to hexadecimal. To do this, right-click the entry in the Monitors view and select **Change representation > Hexadecimal** from the pop-up menu:



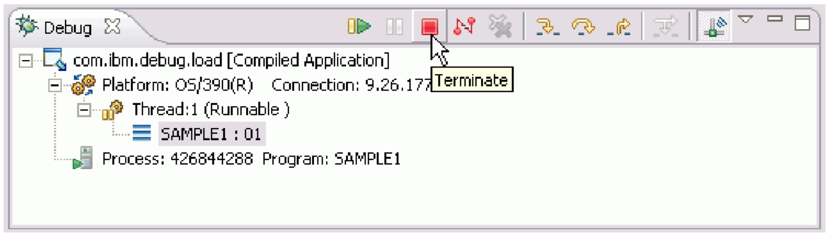
```
days = EQA2302E Not allocated
*charge = EQA2302E Not allocated
rate = EQA2302E Not allocated
CUSTOMERS
  CUSTREC
    CUST-INDEX = 5
    CUSTNAME
      SUB(1) = 'Sprat, Jack                  '
      SUB(2) = 'Quite-Contrary, Mary Mary      '
      SUB(3) = 'Nimble, Jack B.                '
      SUB(4) = 'Pumpkineater, Peter Peter      '
      SUB(5) = 'Muffet, Little Miss            '
    APR
    BALANCE-OWING
      SUB(1) = +0000100.00
      SUB(2) = +0012393.95
      SUB(3) = +0000013.50
```

This will change the value of the entry to a proper signed 7.2 packed decimal 1:



```
days = EQA2302E Not allocated
*charge = EQA2302E Not allocated
rate = EQA2302E Not allocated
CUSTOMERS
  CUSTREC
    CUST-INDEX = 5
    CUSTNAME
      SUB(1) = 'Sprat, Jack                  '
      SUB(2) = 'Quite-Contrary, Mary Mary      '
      SUB(3) = 'Nimble, Jack B.                '
      SUB(4) = 'Pumpkineater, Peter Peter      '
      SUB(5) = 'Muffet, Little Miss            '
    APR
    BALANCE-OWING
      SUB(1) = +0000100.00
      SUB(2) = +0012393.95
      SUB(3) = +0000013.50
      SUB(4) = X'000000100C'
      SUB(5) = -0000200.00
    DAYS-SINCE-PAID
    PENALTY-OWING
```

44. Congratulations! You have completed the tutorial and you can now terminate the debug session. To do this, click the **Terminate** icon in the Debug view. Alternatively, right-click the debug target and you will see other terminate options.



com.ibm.debug.load [Compiled Application]
Platform: OS/390(R) Connection: 9.26.177
Thread:1 (Runnable)
SAMPLE1 : 01
Process: 426844288 Program: SAMPLE1

Summary

In this sample, you have learned how to use WebSphere Developer Debugger for zSeries to perform some basic debug actions on a sample MVS COBOL application.

Notices

Portions based on *Design Patterns: Elements of Reusable Object-Oriented Software*, by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Copyright (c) 1995 by Addison-Wesley Publishing Company, Inc. All rights reserved.

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

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 documentation 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 documentation. The furnishing of this documentation does not give 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, 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 OR CONDITIONS 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.

© Copyright IBM Corp. 2005

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
20 Maguire Road
Lexington, Massachusetts 02421-3112
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this documentation and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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 or 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 may 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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 2000, 2005. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

See http://www.ibm.com/legal/copytrade.shtml.

Printed in USA