CA InterTest™ and CA SymDump® - 11.0

Using the Eclipse UI

Date: 06-Jun-2018

Table of Contents

Synchronized Processing 7

Online Help 8

F1 Support 8

Dynamic Help 9

Demo Source Programs 10

CICS Demo Source Programs 10

Batch Demo Source Programs 10

Understanding Your Workspace 11

Main Window 11

Toolbars 12

Perspectives 13

Views 13

Debugging with the Eclipse UI 16

Connect to a Server 17

- Import CA Testing Tools Server Settings 17
- Add a New Server Connection 17
- Check the Server Status 19

Create a Project 19

Configure the Debug Settings 20

- Configure Debug Settings for CICS Applications 20
- Configure Debug Settings for Batch Applications 20

Select the Source Program to Monitor 22

- How to Use the Outline View 24
- View Monitoring Status 24

Start Program Execution 25

- For Batch Applications 25
- For CICS Applications 25

Analyze Abends 26
Resume Program Execution 26

Abend Analysis with the Eclipse UI 28
Determine the Cause of an Error 29
Change the Value in TASKNUM for COBOL and PL/I Applications 29
Change the Value in TASKNUM for Assembler Applications 30
Breakpoints 31
When to Use Breakpoints 32
Set Unconditional Breakpoints 33
Change Breakpoints - CICS 35
Change Breakpoints 36
How to Delete Breakpoints 38
Bookmarks 38
Add Bookmarks 39
How to Remove Bookmarks 39

Visual Debugger 41
Getting Started with Visual Debugger 41
Statement Trace Visualization 43
Programs Level 44
Paragraphs Level 45
Navigation 45
Visualization of an Entry Point 46
Dynamic Graph Update 46
Example Image Breakpoint at 000028: 46
Example Image Breakpoint at 000034: 47
Navigate Execution Flow History 48
Link with Editor 49
Navigate in the Editor using Graph Elements 50
Navigate through the Graph using the Cursor 51
Zoom and Pan 52
Zoom In and Out 53
Pan the Graph 53
Node Dragging 53
Save and Load Session Data 54
Save a Debugging Session 54
Load a Debugging Session 55

Using the Eclipse UI

This section describes how to use the Eclipse UI to debug CICS and batch applications. Use the Eclipse UI and this document to walk through the process and procedures that follow.

Tip: This section contains tutorial instructions that step you through debugging a program with the Eclipse UI. When you see instructions that prompt you to make specific changes a debug session, use the information as an example, or complete the tutorial by following the steps as instructed. The tutorial uses the default CA InterTest for CICS library names. See Demo Source Programs (see page 10) for the names of the demo programs.

To perform the steps in this tutorial, you must know the following information. If you do not already know this information, or if the default demo program names were changed during installation, contact your CA InterTest administrator.

From your IBM installation

- User ID
- z/OS host name

From CA InterTest

- Path to the file containing Testing Tools server settings to be imported. If the application was installed properly, you have a set of servers that are properly configured for your site.
- CA InterTest for CICS PROTSYM data set names

For the CICS demo sessions, you also need the following information:

- CICS host name
- Path to the file containing CA InterTest server settings to be imported

Synchronized Processing

The application searches the symbolic files for the programs that you specify to be monitored for testing and tries to match a symbolic file with the same date and time as that of the load module. If a match is not found, the Select Symbolic dialog opens. If this dialog appears, do one of the following:

- Select the version that you want the mainframe debugger to use and click Use.
- If you do not want to monitor the file, select Do Not Monitor.
- For batch applications, you can click Abend Stop to end the debugging session.

Use this feature to maintain synchronized processing at all times.

Online Help

CA InterTest provides extensive online documentation and context-sensitive help. You can request online help anytime while using CA InterTest by clicking the Help menu and selecting the following options as needed.

- **Help Contents**
  Opens an external Help browser window and displays a top-level link to the CA InterTest online documentation, which you can expand to display subsets of the documentation.

- **Search**
  Activates the CA InterTest online Help view and displays the Search panel in which you can enter text that you want to search for. Click Go to display all topics on the online documentation in which the search text appears.

- **Dynamic Help**
  Displays the help data as it is dynamically displayed to reflect the context of the view that is in focus. For example, when the focus is on Breakpoints, Dynamic Help will only display those topics relevant to breakpoints.
  Note: The dynamic help feature is not supported in the Program Listing area pop-up menus, window controls such as menu items or buttons, or toolbar buttons.

- **Cheat Sheets**
  Opens pages to guide you through some of the processes. Each cheat sheet is designed to help you complete a specific task, and lists the sequence of steps required to help you achieve that goal.

- **Key Assist**
  Opens the Key Assist window where you can see shortcut key combinations.

- **Update InterTest**
  Opens the Install/Update dialog from which you will be able to install updates of the currently installed features, or new features.

- **About CA InterTest**
  Displays the About CA InterTest dialog, which provides information on the software release number, plug-in details, and configuration details.

F1 Support

You can access online Help for most (but not all) window controls by placing the focus on the dialog or control and pressing F1.

If the control is located in the CA InterTest main window, the Help view is activated automatically and a keyword search targeting text in the online documentation related to the control is performed. When the search is complete, links to topics related to the keyword search are displayed in the Help panel.

If the control is located on a dialog started by CA InterTest, an external Help window is activated. The same keyword search is performed and links to topics related to the keyword search are displayed in the Help panel.

Dynamic Help

When the CA InterTest Help view is active in CA InterTest as you make each view active, the help data displayed is dynamically updated to mirror the view that is in focus.

Demo Source Programs

CA InterTest for CICS installs demo programs to use for this tutorial that are designed for use with the CA InterTest for CICS Eclipse UI tutorial. Use the compile procedures created during installation to compile a demo program, populate the PROTSYM, and link the program.

Note: The default names for the demo programs for each type of application may have been changed by your CA InterTest for CICS administrator during installation.

CICS Demo Source Programs

The demo CICS source programs are as follows:

Name of Program	Language Type	Transid
COBDEMO	Enterprise COBOL	DEMC
DB2DEMO	Enterprise COBOL	DEMD
PLIDEMO	PL/I	DEMP
ASMDEMO	Assembler	DEMA

Batch Demo Source Programs

The demo source programs for batch applications are as follows:

Type of Program	Name of Program
Assembler	CAMRASM
COBOL	CAMRCOBB
PL/I	CAMRPLI

Understanding Your Workspace

Contents

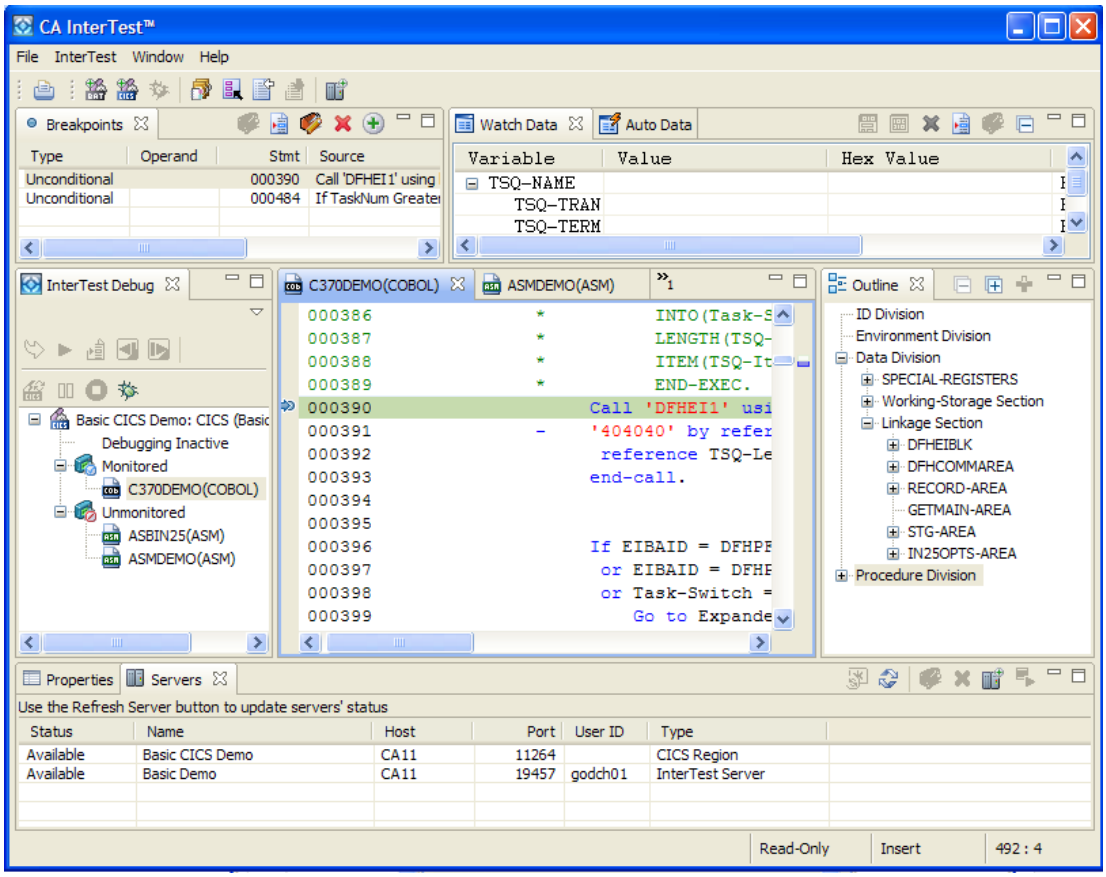
- Main Window (see page 11)
- Toolbars (see page 12)
- Perspectives (see page 13)
- Views (see page 13)

Main Window

The CA InterTest and CA SymDump Eclipse UI contains a group of views that provide all the functions you need to debug your programs.

A *view* is a visual component within the application window. It is typically used to navigate a hierarchy of information or display properties for a program listing, which displays the program being debugged. Modifications made in a view are generally done with dialogs. A *perspective* is a collection of views that pertain to a single product.

The following illustration shows the main window in the CA InterTest perspective with several views open.



CUI--Main Window--SCR

Usually, only one instance of a particular type of view may exist within the main window. Views can appear by themselves or stacked with other views in a tabbed notebook. To activate one of these views, click the appropriate tab. Only one view is active at one time, with the exception of the Program Listing area, where several imported program listing files can be open at one time. The listings behave like sub-views that are grouped together (or stacked) in the Program Listing area. You can change the default window layout by opening views, closing views, or docking views in different positions.

Toolbars

There are two kinds of toolbars in the CA InterTest and CA SymDump Eclipse UI:

- Main toolbar
- View toolbar

The *main toolbar* is displayed at the top of the main window directly beneath the menu bar. The contents of this toolbar change based on the active view. You can find explanations of all the buttons on the toolbar in the online help.

There are also individual *view toolbars*, which appear in the title bar of a view. Actions in a view's toolbar apply only to the view in which they appear. The following example shows the Bookmarks view toolbar:

CUI--Breakpoints Tab--ICO

> **Note**: Some view toolbars include a Menu button, shown as an inverted triangle, that contain actions for the view.

Perspectives

The Eclipse UI has one perspective for CA InterTest and another for CA SymDump. You can switch from one product to the other without closing the Eclipse UI, but you must always have at least one perspective open.

To open a perspective, select Window, Show perspectives. When the Open Perspective dialog appears, click the perspective that you want to open. The main window updates to show the perspective that you selected..

Views

The CA InterTest and CA SymDump Eclipse UI supports the following types of views:

- **Auto Data**
  Displays variables and their contents at a particular breakpoint.
- **Bookmarks**
  Displays all anchors (bookmarks) placed on a specific line of code.
- **Breakpoints**
  Displays all breakpoints (enabled and disabled) that are set in the vertical ruler of the Program Listing area.
- **Call Trace**
  Displays a list of the called subprograms in a load module based on the current breakpoint. This is a CICS only view.
- **Console**
  Displays read-only standard output messages from CA InterTest.
- **Error Log**
  Displays all warnings and errors generated by the common user interface. This is used by CA support and you can export this log when reporting an error.

- **Help**
  Displays the related help topics for the highlighted view.

- **InterTest Debug**
  Displays the monitored and unmonitored programs.

- **Outline**
  Displays an outline of the program that is currently open in the Program Listing area and lists the structural elements.

- **Properties**
  Displays a list of attributes, if any, related to another active view. Not all views have information to display in this view.

- **Register Data**
  Keeps track of general register values during a debugging session for Assembler programs.

- **Servers**
  Displays the servers that have been defined for CA InterTest to communicate with the mainframe.

- **Session Data**
  Keeps track of session-related information during a debugging session for CICS programs.

- **Statement Trace**
  Displays a trace of previously executed statements.

- **Tasks**
  Displays the tasks available for the currently active program.

- **Watch Data**
  Keeps track of variables and other program-related information during a debugging session.

Use views to navigate a hierarchy of information (such as variable values), show breakpoints, bookmarks, console output, and various other activities.

The application saves any changes you make in a view across instances of the Eclipse UI.

To open a view, select Window, Show views, and then click the view that you want to open. The selected view opens in the main window.

To close a view, click on the X on the view's tab.

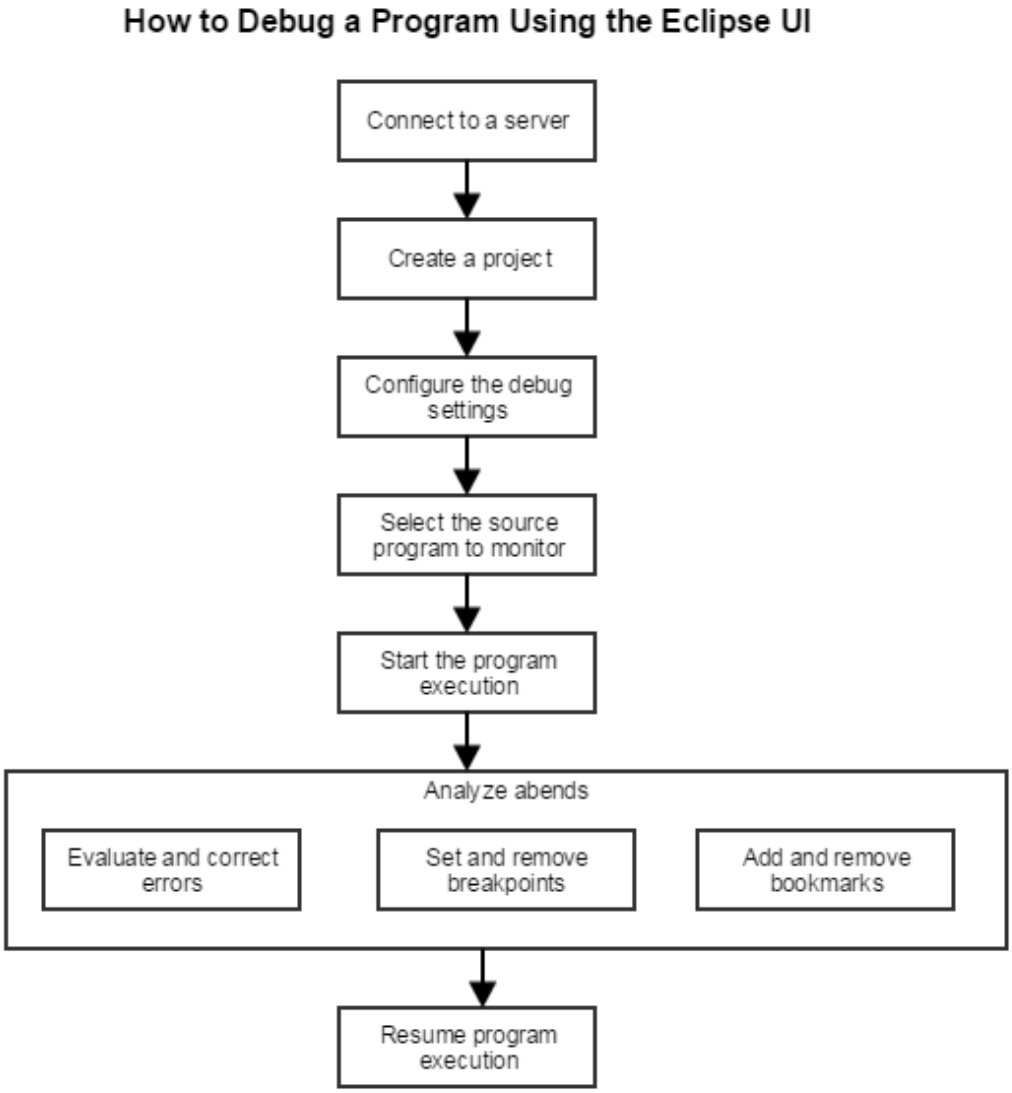
You can perform the following actions with the buttons on the Bookmark's toolbar, as described in the following table:

Button	Description
 CUI--Configure Program Settings--ICO	Configures the settings for the program name
	Shows the position in the program listing where the breakpoint is set

Button	Description
 CUI--Position in File--ICO	
 CUI--Configure Selected Breakpoints--ICO	Configures a selected breakpoint
 CUI--Delete Icon--ICO	Deletes a selected breakpoint.
 CUI--Add A Global Breakpoint--ICO	Adds a global breakpoint
 CUI--Minimize Button--ICO	Minimizes the view.
 CUI--Maximize Button--ICO	Maximizes the view.

Debugging with the Eclipse UI

This article describes how to use the Eclipse UI to debug a program.



To debug a program using the Eclipse UI:

1. Connect to a server (see page). (see page 17)
2. Create a project (see page 19). (see page 19)
3. Configure the debug settings. (see page 20)
4. Select the source program to monitor (see page 22).

5. Start the program execution (see page 25)
6. Analyze abends. (see page 26)
7. Resume program execution (see page 26)

Connect to a Server

Before you begin debugging a program, you must connect to a server. You can import server setting and you can add new server connections to the Eclipse UI.

Import CA Testing Tools Server Settings

Connect to a Testing Tools server before, start a debugging session. The GUI component runs on Windows and the debugging session runs on z/OS. This server provides the communication between the Eclipse UI and the mainframe platform.

You may need to import the Testing Tools server settings the first time you start the Eclipse UI component or whenever your CA InterTest administrator updates any of the server settings.

If the application was installed properly, you have a set of servers that are properly configured for your site.

Follow these steps:

1. Start the CA InterTest Eclipse UI.
   The CA InterTest main window opens.
2. Click File, Import.
   The Import dialog opens.
3. Click Next.
   The Import Server Settings dialog opens.
4. Type or browse for the file path to the file containing the server settings. Obtain this information from your CA InterTest administrator.
5. Click Finish.
   You are connected to the specified server.

Add a New Server Connection

If you have not already established a connection to the server that you want to use, you can add a server connection using the Eclipse UI. You must have the host and port details of the server to which you want to connect.

Follow these steps:

1. Select InterTest, Add New Server.
   The New Server Connection dialog opens.

2. Enter the details for the new server connection.
   The following field values are required to create a new server connection:

   - **Name**
     Specifies the name of the server connection. You can use uppercase and lowercase alphanumeric characters, and embedded spaces when entering a value in this field.

     Tip: For the purposes of this tutorial, specify the name **Basic Demo** in this field.

   - **Host**
     Specifies the z/OS host name assigned by your installation. This is the host name used for TSO or CICS logon.

   - **Port**
     Specifies the TCP/IP port number of the server. If there is a default port number displayed, verify it is correct. The port number is updated in the Preferences dialog.

     - For a CICS region -- This field must be the port number configured in the CICS region.

     - For a Testing Tools server -- This field must be the port number that is configured for the Testing Tools server running on z/OS UNIX System Services (USS).

   - **Use as Default Server**
     Specifies that this server is selected when you access the Batch Link Queue or Schedule Table.

   - **CICS Region**
     Defines a connection that is used when a CA InterTest for CICS for CICS debug session is initiated.
     Select this box to initiate a CICS debug session. This action displays the CICS Security Enabled check box. If you select this box, you can secure your connection by using a user ID and password.
     Clear the CICS Region check box to define a server connection.
     Default: Off

   - **(Optional) User ID**
     Specifies the user ID for logging on to a z/OS system. Enter your user ID in this field.

   - **(Optional) Save Password**
     Specifies whether you want to save the password. If you save the password, the application preloads the saved password into the Logon dialog whenever you are prompted to enter your user ID and password. Until the password is deliberately reset or a logon fails, you are prompted for a user ID and password only when you first use a server.
     Check this box to save your password.

3. Click Finish.
   The new server is created and added to the list of servers in the Servers view. The Servers view shows the details of the new server.

Check the Server Status

The Server view displays the status of the servers you are connected to so that you can see if the server is active.

Follow these steps:

1. Click Window, Show Views, Servers, on the CA InterTest main window toolbar if the Servers view is not already visible.
   The Servers view opens, displaying the name of the server and its status.
   The server has one of the following statuses:

   - **Available**
     The GUI is connected to a mainframe server that is running.

   - **Not Available**
     The GUI failed to connect to the server and is not running. The reason for the failure is noted next to the server status.

2. If the Status column is blank, click the Refresh Server button  on the Servers view toolbar.
   The system may take several seconds to respond.

3. If the status remains blank or Not Available, contact your CA InterTest administrator.

Create a Project

After you are connected to a CA Testing Tools server, create a project in which to debug your program.

Follow these steps:

1. Click the Create Project button  on the CA InterTest main window toolbar.
   The New CICS Project dialog opens or New Batch Project dialog opens, depending on the demo session.

2. Specify a name for your project in the Project Name field.

   Tip: Specify **Basic Demo** in the Project Name field for the Eclipse UI tutorial. The rest of this procedure uses this demo value.

3. Check the Use default location box if it is not already checked.

4. Select Basic Demo from the InterTest Server drop-down list.

5. For CICS applications, select Basic Demo from the CICS Region drop-down list.
   Note: Do not select Import files once project is created box.

6. Click Finish.
   The InterTest Debug view displays your project name, project type, the server name, the status indicator, and two empty folders: Monitored, and Unmonitored.

Configure the Debug Settings

Before you execute a program, set the debug settings for your project. The debug settings dialog is different for CICS and batch applications. This dialog is also displayed when you start the program execution.

Configure Debug Settings for CICS Applications

Follow these steps:

1. Right-click the project name on the InterTest Debug view, and select Configure Session Debug Settings.
   The InterTest Debug Settings dialog opens.

2. Ensure that the following fields are completed:

   - **InterTest Server**
     Displays the Testing Tools server name.

   - **CICS Region**
     Displays the CICS region name.

3. Click Finish.
   The debug settings are configured.

Configure Debug Settings for Batch Applications

Tip: The specific values in this procedure apply to the demo program tutorial.

Follow these steps:

1. Click the project name on the InterTest Debug view, right-click, and select Configure Session Debug Settings.
   The InterTest Debug Settings dialog opens with the main tab in view.

2. Click the InterTest Server drop-down list, and select Basic Demo.

3. Select the PROTSYM tab.

4. Click Add, and add CAI.PROTSYM as the PROTSYM data set name, or the data set name provided by your CA InterTest administrator, in the Add Dataset Name dialog. Click OK.
   The data set name appears in the PROTSYM Datasets area.

5. Click the JCL Settings tab.
   The InterTest Debug Settings dialog opens.

6. Complete the following fields:

   - **InterTest Server**
     Displays Basic Demo. If Basic Demo is not showing, select it in the box.

   - **Conversion JCL Dataset(Member Name)**
     Describes the mainframe data set name and member name. Type **CAI.CAVHJCL(DEMOJCL)** or a data set name provided by your CA InterTest administrator.

   - **Dataset Type**
     Describes the type of data set. Select IBM PDS, or the data set type appropriate to the one provided by your CA InterTest administrator.

   - **Job Step Name**
     Describes the name of the job step. Type **Step1**.

   - **PROCLIB List**
     Leave empty.

   - **Converted/Submit JCL Dataset(Member Name)**
     Describes the output data set where the converted JCL resides. Type **HLQ.MYJCL (BATCHDMO),** where HLQ is the high level qualifier you choose, or any allocated PDS.

7. Click the Convert JCL button.
   The converted JCL is placed in HLQ.MYJCL(BATCHDMO).

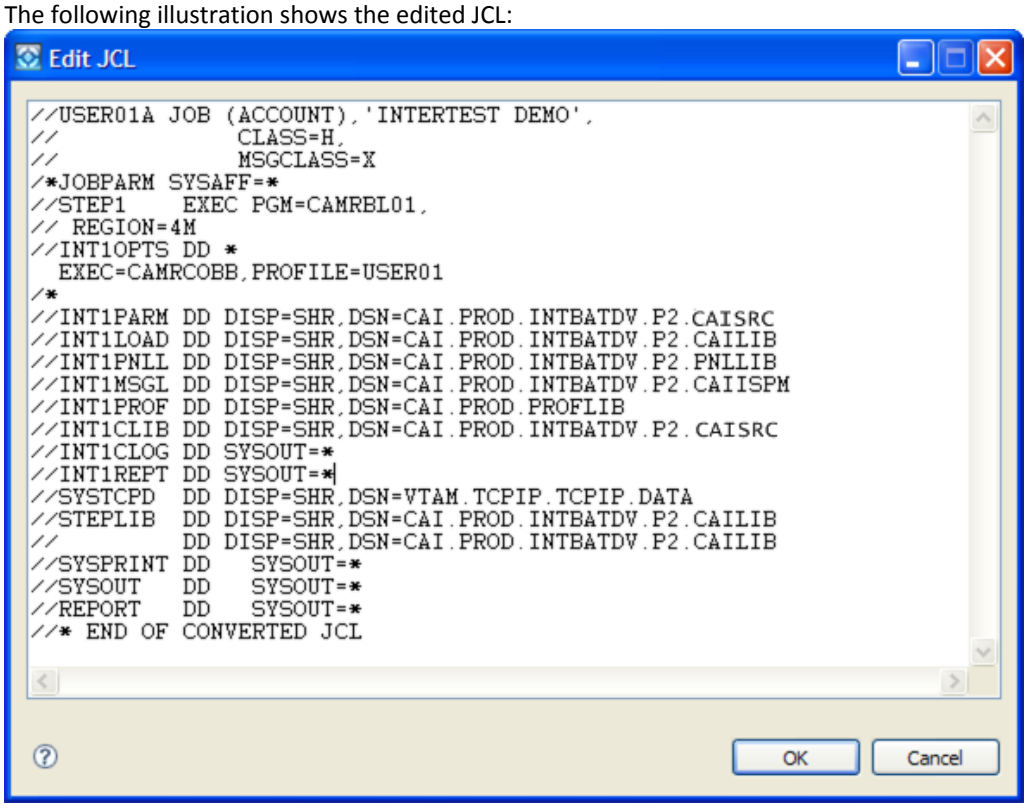
8. After the JCL conversion is complete, click the Edit JCL button to edit the converted JCL.
   The Edit JCL dialog opens.

9. Make the following changes to the Edit JCL dialog:

   - Change the job card to conform with your installation standards.

   - Change PROFILE=YOURID to PROFILE=(your TSO User ID).

The following illustration shows the edited JCL:



10. Make sure the data set you are editing is not being used in TSO or CA Roscoe.

11. Click OK.
    The Edit JCL dialog closes and you return to the InterTest Debug Settings dialog.

12. Click Finish.
    You receive a message indicating that the file has been saved successfully. Click OK. You return to the InterTest main window.

Select the Source Program to Monitor

After configuring your session, specify the program that you want to monitor. As part of monitoring a program, the application automatically prevents all abends during execution.

Follow these steps:

1. Verify that the program to be debugged has been compiled and post-processed into the PROTSYM files.

2. Click the InterTest Debug view so that it is the active view.

3. Click the project name on the CA InterTest Debug view.
   The project name is highlighted.

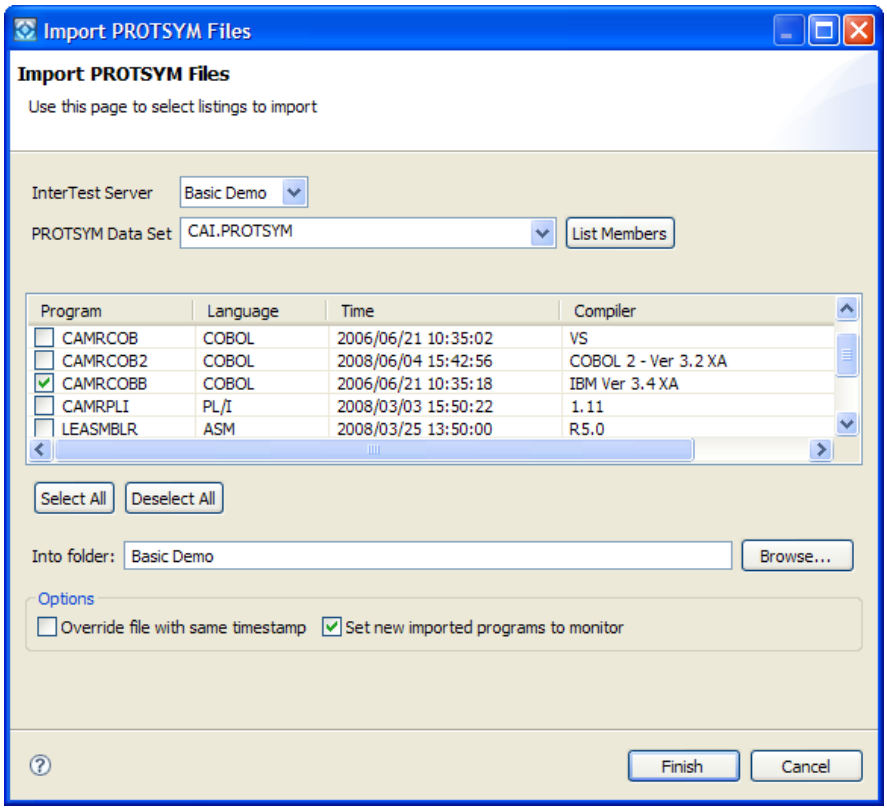
4. Click the Import PROTSYM Files button  on the CA InterTest main window toolbar.
   The Import PROTSYM Files dialog opens.

5. Enter the following information:

   - **InterTest Server**
     Defines the Testing Tools server. For the tutorial, select Basic Demo.

   - **PROTSYM Data Set**
     Defines the fully qualified data set name of the PROTSYM VSAM file.
     Type **CAI.PROTSYM,** or substitute CAI.PROTSYM with the DSN provided by your CA InterTest administrator.
     If you receive an error about failure to retrieve a member list, check with your CA InterTest administrator.
     Note: You may be prompted to log on to your mainframe TSO here, if you have not already done so.

6. Click List Members.
   The table is populated with the programs in the data set you chose.



Import PROTSYM Files

7. Scroll down through the table until you reach the program that you want to use. For the names of the demo programs to use in the tutorial, see Demo Source Programs (see page 10) . Select the program by checking the box next to it.

8. If the Into folder box is empty, click the Browse button, and select the Basic Demo folder to import the program into.
   The name Basic Demo appears in the Into folder box.

   Note: The CA InterTest administrator may have changed the name of the sample programs when installing the application. If you do not see the name of the demo program that you want to use, check with your CA InterTest for CICS administrator to find out the correct name.

9. Ensure that the two check boxes in the Options section are selected.

10. Click Finish.
    The demo program displays in the InterTest Debug view in the Monitored Programs folder.

11. Double-click the demo program name on the InterTest Debug view.
    The program listing displays in the Program Listing area, with an outline of the program in the Outline view.

How to Use the Outline View

The Outline view displays an outline of the program that is currently active in the Program Listing area and lists the structural elements. This makes it easier to display any section of your program. The contents of the Outline view are specific to the active program listing. The Outline view contains a top level node for each action.

- To expand a node, click the plus sign (+) next to the text.
- To collapse a node, click the minus sign (-) next to the text.

Note: Selecting a variable or label name repositions the program listing to the line containing the name. If the location is collapsed, the area is automatically expanded.

View Monitoring Status

The InterTest Debug view lists the programs in a tree that identifies them as monitored or unmonitored. You can view the programs under these categories by clicking the plus (+) to expand the Monitored and Unmonitored nodes.

Start Program Execution

The process to start the program execution is different for batch and CICS applications. Follow one of these procedures to begin debugging your program.

For Batch Applications

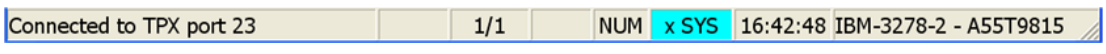
Follow these steps:

1. Select the program that you want to monitor under the Monitored node in the Debug view.
2. Click the green bug button  on the toolbar.
   The batch job is submitted and the initial intercept is displayed.
3. Click the green arrow button to allow the program to run.
   The program will run until it encounters an abend.

For CICS Applications

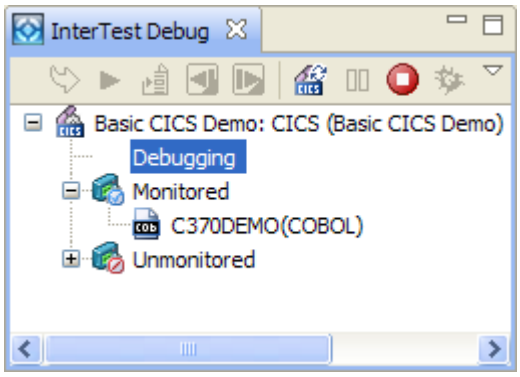
Follow these steps:

1. Under the Monitored node on the InterTest Debug view, select the program that you want to monitor and click the green bug button  on the toolbar.
   The InterTest Debug Settings dialog opens.
2. Ensure that the debug settings are defined as you want them.
3. Click Connect.
   CA InterTest notifies CA InterTest for CICS that your program is ready for debugging.
4. Go to the mainframe CICS display (green screen), and type in the transaction identifier of the demo program as listed in Demo Source Programs (see page 10).
5. Press Enter.
   The Welcome screen appears.
6. Press Enter on this CICS mainframe screen.
   The demo program begins execution. Your CICS session will indicate that it is in xSystem state:



Xsystem state indicator

1. Return to the CA InterTest Eclipse UI to continue with your debugging session. Notice that the InterTest Debug view now displays the debug status as Debugging.



Analyze Abends

When a program encounters an error during execution, an abend occurs. See Abend Analysis with the Eclipse UI (see page 28) for more information about addressing abends that occur during program execution.

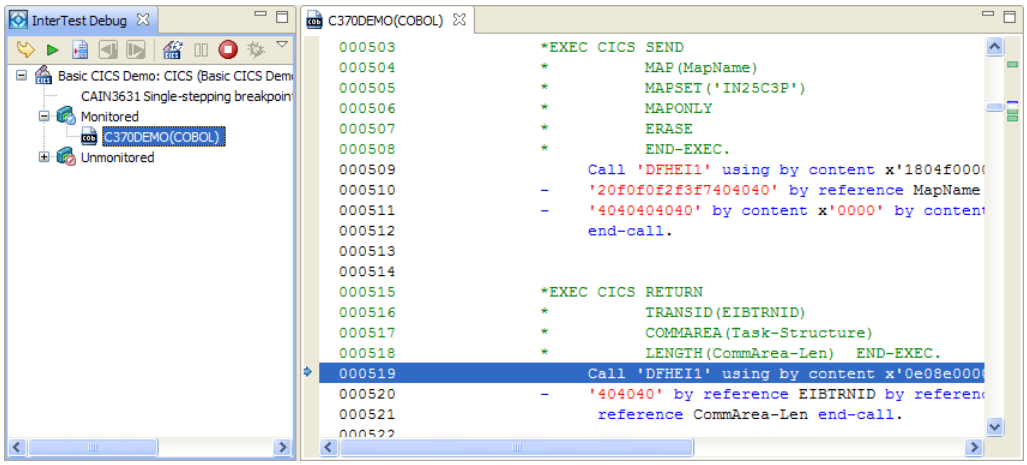
Resume Program Execution

Resume the program execution after evaluating and resolving an error that the program identified or after the program has stopped at a breakpoint.

Follow these steps:

1. Click the Step Once button  on the InterTest Debug view toolbar to continue testing by resuming program execution.

2. Program execution stops at the next statement. The debug status in the InterTest Debug view indicates that the program is at a single-step breakpoint and that shows the value that you updated in resolving the previous abend.
   The following example shows the results of this step for the CICS COBOL demo.



Step once results

3. Click the Resume Debugging button  on the InterTest Debug view toolbar.
   The demo program halts at the unconditional breakpoint you set previously in this tutorial.
   The InterTest Debug view window updates to reflect the reason for the stop

4. Click the Resume Debugging button again.
   The program continues until no further abends, breakpoints, or bookmarks are encountered.

5. When the program completes, do one of the following actions to end the debugging session.

   - For batch applications, the demo program displays the Debugging dialog to indicate that the program is complete. Click OK to terminate the debugging session. The tutorial is finished.

   - For CICS applications, the mainframe CICS screen displays the END DEMO SESSION screen. Press Clear or Enter on the mainframe CICS screen to end the CICS debugging session and then complete the next two steps.

6. In the Eclipse UI, click the Cancel Debug Session button  on the InterTest Debug view toolbar.
   The Cancel Debug Session dialog opens.

7. Click the Disconnect and Remove Breakpoints button, and then click Finish to terminate the connection with the server.
   The debugging status now shows Debugging Inactive in the InterTest Debug view.

Abend Analysis with the Eclipse UI

When a program encounters an error during execution, an abend occurs and stops the program. This article describes how to identify and fix the cause of an abend before resuming the program execution.

At the abend, the following events occur:

- The appropriate program listing causing the error is activated (if the program was not displayed earlier).
- The program positions itself at the statement causing the error.
  When the program stops, we say that it halts the program at a breakpoint. This is done automatically when an abend occurs, or the application halts a program at a breakpoint that you have previously set.

When an abend occurs:

- A small blue arrow appears in the ruler to the left of the line causing the error, and the line is highlighted.
- The status in the InterTest Debug view now displays a diagnostic message. This message explains that the error was caused by improperly formatted data.

> **Tip:** In the tutorial, the demo program detects an error at the ADD instruction. Execution of that ADD instruction triggers an abend.

Once a program is stopped, you can use the CA InterTest testing and debugging facilities to do the following tasks. When debugging your own programs, you will typically perform one or more of these activities.

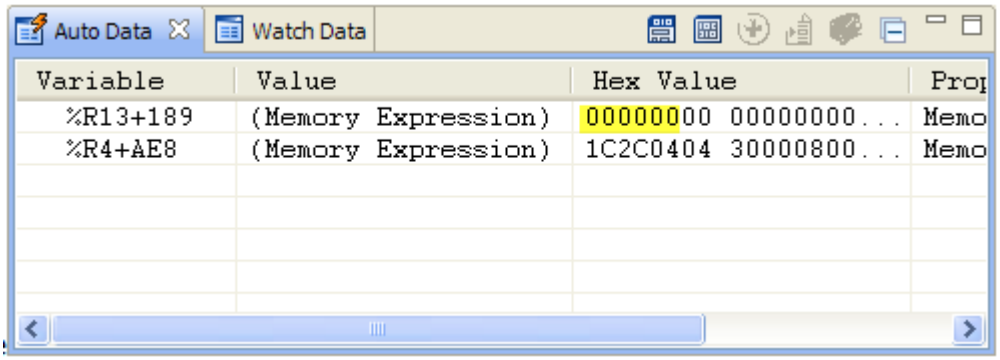
- Examine the source listing.
- Examine and modify main and auxiliary storage (memory) to detect and correct errors.
- Set and remove breakpoints and bookmarks.
- Examine a program's statement trace.
- Keep data items in a watch view to observe changes in their values.
- Abend your task with or without a dump.
- Go around a problem by resuming program execution from a location other than the one at which the program is currently stopped.
- Execute the program in single-step mode; that is, the program executes one verb and then stops.

Determine the Cause of an Error

Use the Auto Data view to see the values of the variables that are associated with a statement. Use the Watch Data view to see the value of data items that you want to watch.

Example:

In this example, an Assembler program has encountered an error at the TASKNUM statement. The value of TASKNUM is displayed as an expression (%R13+189) in the Auto Data view. The left column shows the base + displacement for the name of the data item. In the columns to the right are the field's display value. Assembler programs show as Memory Expression, and hex value at that offset.



Look at the contents of TASKNUM. Confirm that the value stored in TASKNUM is not a valid packed decimal value by examining its current value in the Auto Data view; that is, its value prior to the execution of the AP statement that triggered the ASRA. Instead, it contains low-values (binary zeros - the three highlighted bytes). Assembler does not allow you to add a packed decimal value (=P'1') to a field initialized with low values.

Change the Value in TASKNUM for COBOL and PL/I Applications

You have identified and confirmed the cause of the problem. Now you want to fix it.

Follow these steps:

1. Right-click the variable TASKNUM in the Auto Data view. Click Change Value.
   The Data View Value dialog opens.

2. Select the Display value button.

3. Change the value of TASKNUM displayed in the box to a numeric zero (type a numeric zero).

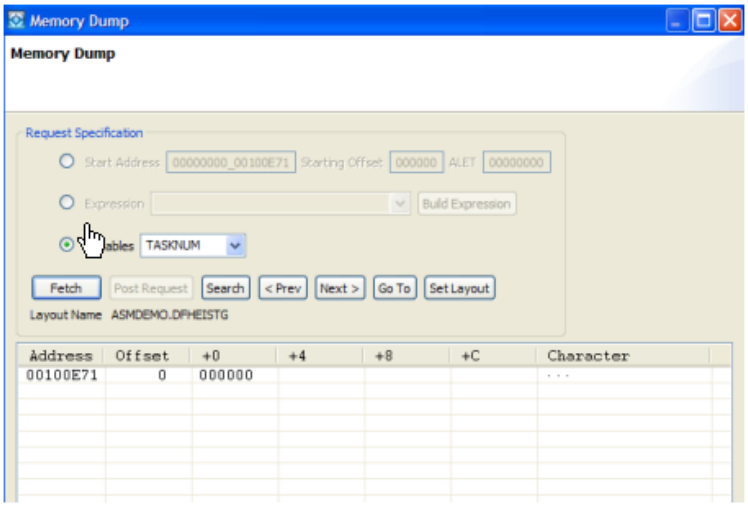
> **Note**: You do not have to know the type of data (binary, packed, and so on) or the length of TASKNUM. You can change the contents of a field by overtyping the desired bytes.

4. Click OK.
   TASKNUM now contains a packed decimal zero.

Change the Value in TASKNUM for Assembler Applications

Follow these steps:

1. Right-click one of the following variables in the Auto Data view
   - %R13+189 - for CICS applications
   - %R13+634 - for batch applications
2. Click Memory Dump.
   The Memory Dump dialog opens and displays the memory address for TASKNUM in the Address column. To the right under offset +0 the hex value displays, to the far right is the character value. The first three bytes (highlighted yellow) are the value of TASKNUM.
3. Click the Variables button, type TASKNUM, and click the Fetch button.
   The Memory Dump dialog opens, and the three bytes of TASKNUM display.



4. Overtype 000000 with 00000C, and press Enter.
5. Click Close.
   The Auto Data view now displays TASKNUM with a packed decimal zero.

Breakpoints

Contents

- When to Use Breakpoints (see page 32)
- Set Unconditional Breakpoints (see page 33)
- Change Breakpoints - CICS (see page 35)
- Change Breakpoints (see page 36)
- How to Delete Breakpoints (see page 38)

A *breakpoint* is an intentional stopping or pausing place in a program, put in place for debugging purposes.

When you open or activate a program listing in the Program Listing area, the breakpoints added to that program are marked with a blue circle icon. When the mouse hovers over a blue circle, the following details of a breakpoint are displayed:

- Breakpoint type
- Program name
- Statement number (in square brackets)
- Source code text

There are six types of breakpoints.

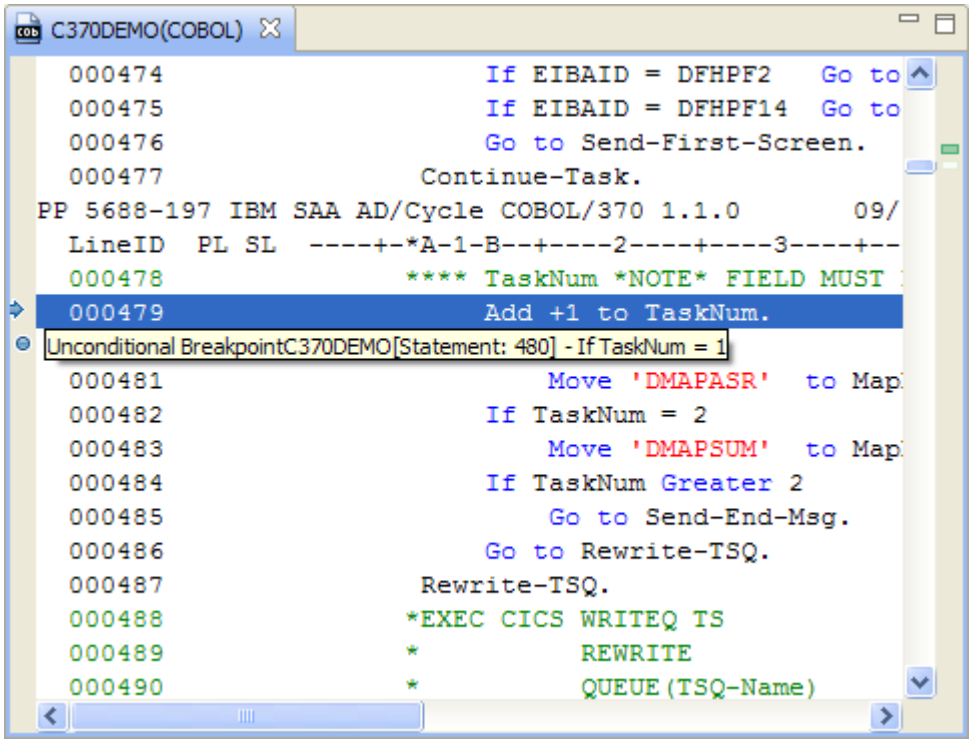
Automatic

The program stops because CA InterTest for CICS detected and prevented an error. When a program is stopped at an automatic breakpoint, you can either correct the error or go around it. You set all other breakpoints.

- **Conditional**
  The program stops at the location you specify if a condition is met. Optionally, conditional breakpoints can be set to stop at *any* instruction if a condition is met.
- **Request**
  The program stops at every CICS command or macro, at certain CICS commands or macros, or at calls to PL/I, DB2, or software.
- **Single-step**
  The program stops after executing one or more verbs.
- **Unconditional**
  The program stops at the location you specify, just before the statement is executed.
- **Variable Change**
  The program stops at any location if the value of a specified variable has changed. This is a special type of conditional breakpoint.

Example:

In this example, execution stops just before the statement Add +1 to TaskNum is executed. A blue arrow points to that statement. This illustration displays the information when the mouse hovers over a blue circle.



CUI--Bpt Hover Detail COBOL CICS--SCR

When to Use Breakpoints

You can set breakpoints at any time -- before you execute the program and when the program is stopped. Although where you decide to set breakpoints depends on the specifics of your program, you might want to set breakpoints in the following places:

- At the beginning of the Procedure Division, so when the program executes you can set additional breakpoints
- At paragraph names, so you can examine the contents of variables at the start of sections
- Before a call, so you can dynamically control the program path
- At each location named in an EXEC CICS HANDLE CONDITION, so you can verify error handling

Set Unconditional Breakpoints

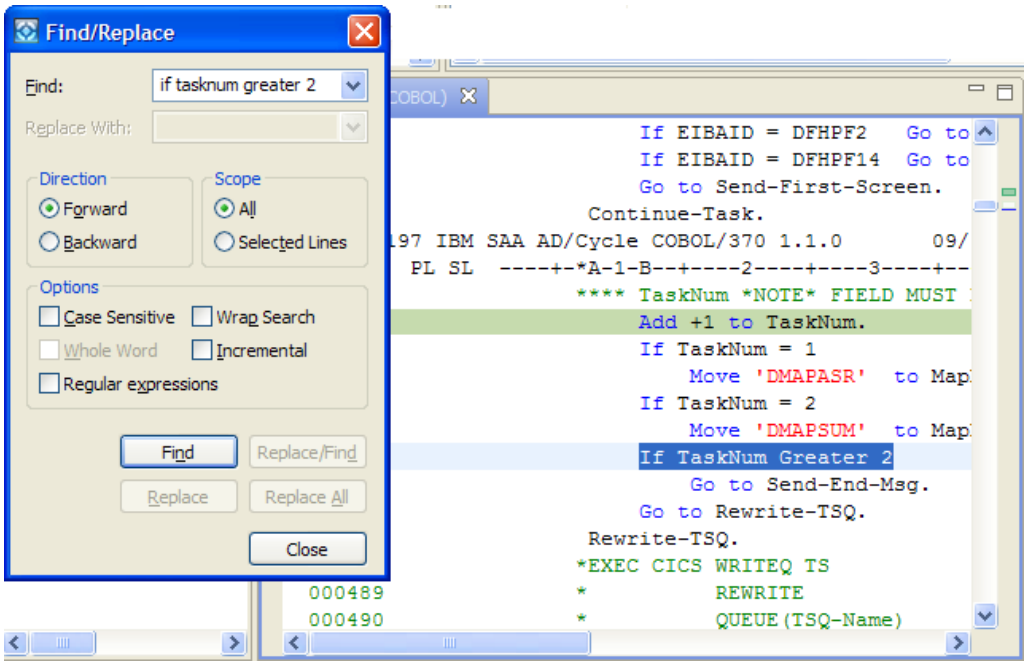
At an unconditional breakpoint, the program stops at the location you specify, just before the statement is executed. Let us see how to set a breakpoint at the following IF statement. You set unconditional breakpoints directly on the Program Listing area.

Follow these steps:

1. Click the demo COBOL program listing on the Program Listing area, so its name is highlighted in its tab.

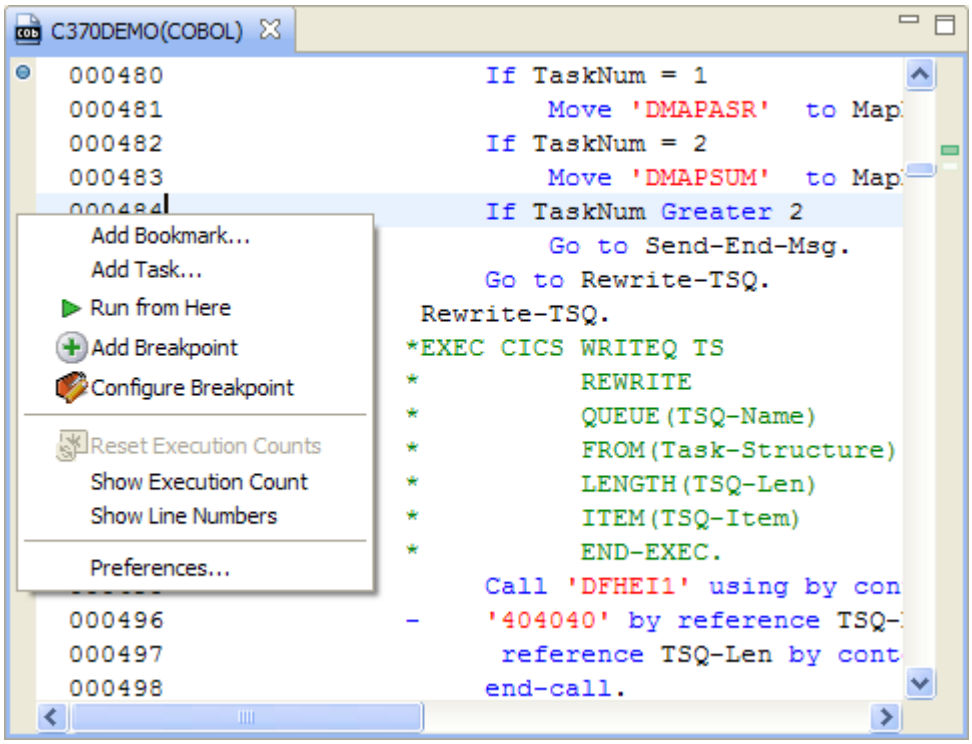
2. Press <Ctrl> + F on your keyboard.
   The Find/Replace dialog opens.

3. Type **If Tasknum Greater 2** in the Find box, and click Find.
   The demo program positions itself so that the line you searched for appears in the program listing.



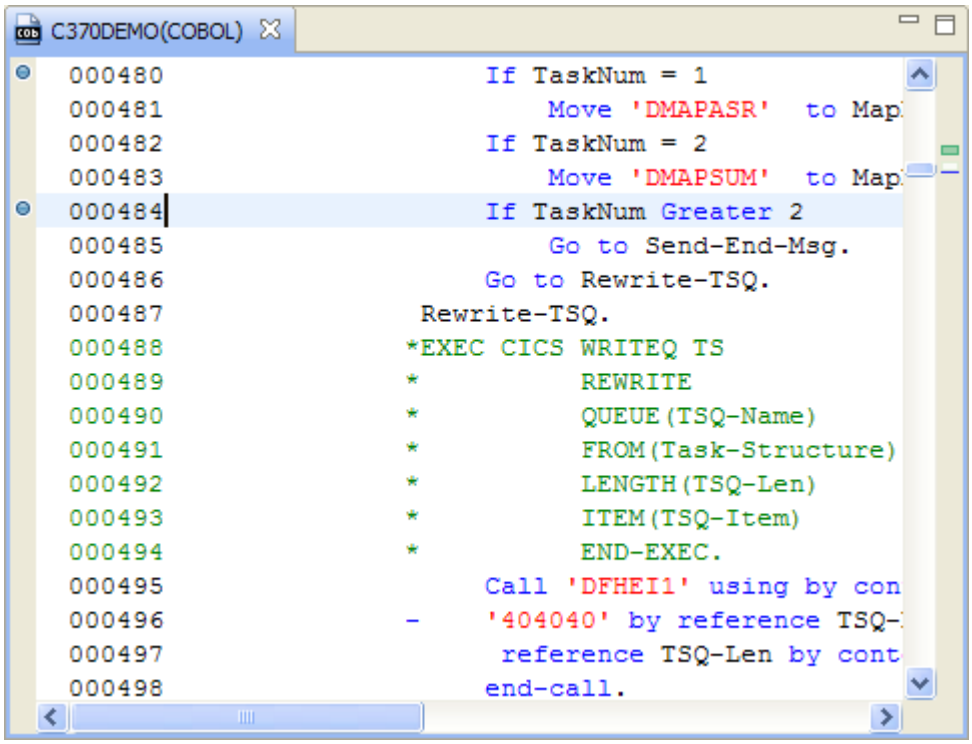
CUI--Add Breakpoint--COBOL--SCR

4. Right-click the ruler to the left of the line you have selected (statement 484), and select Add Breakpoint.



CUI--Unconditional Bpt CICS COBOL--SCR

A blue circle appears to the left of the ruler, and a blue mark appears in the right-side vertical ruler. By default, the Add Breakpoint dialog adds an unconditional breakpoint if the listing line is within the Procedure Division, and a variable change breakpoint if the line is within the Data Division.



CUI--Unconditional Breakpoint CICS COBOL2--SCR

Note: You cannot add a breakpoint in a comment line or outside the source code section.

Change Breakpoints - CICS

The process for changing breakpoints is different for CICS and batch applications. The following procedure describes how to change a breakpoint in a CICS application.

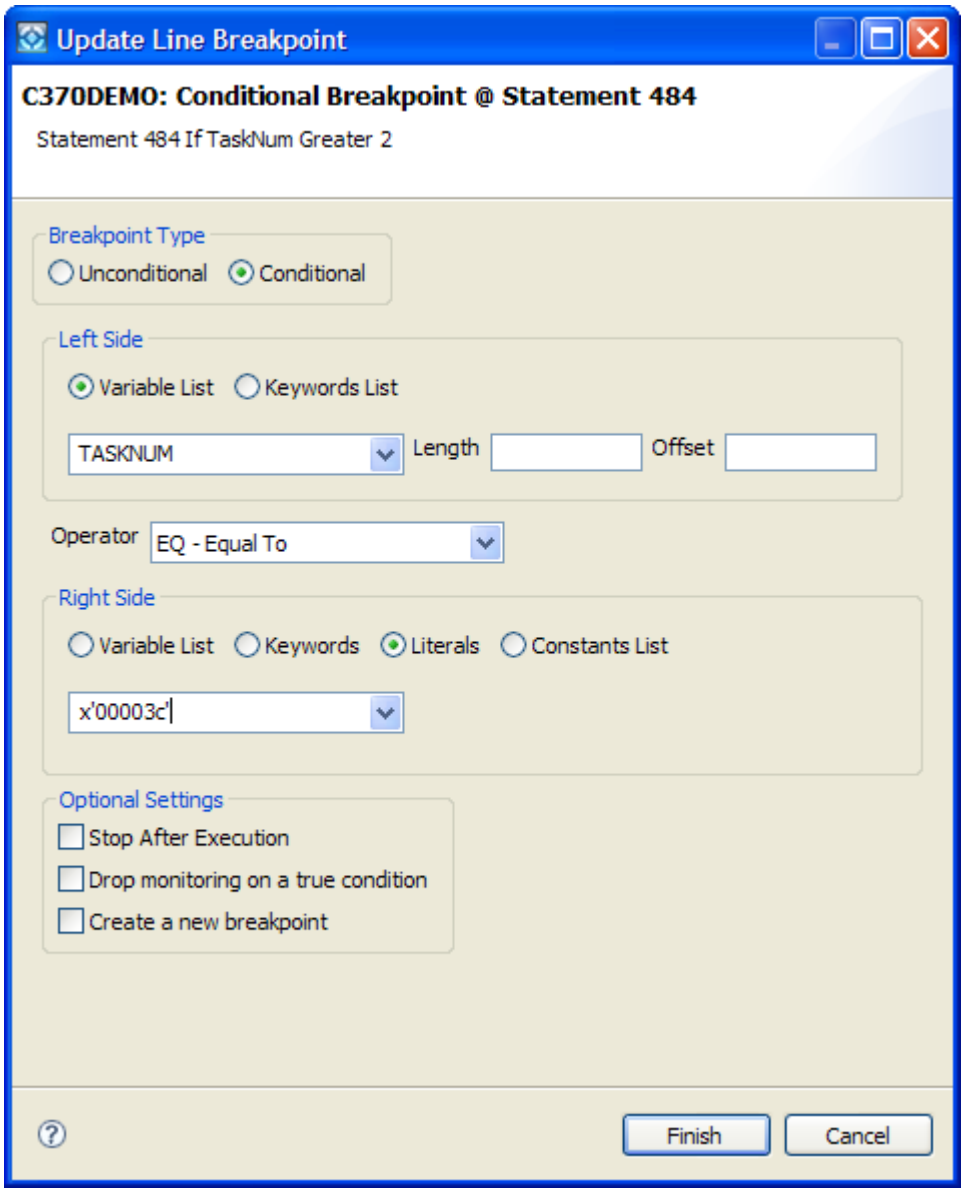
Follow these steps:

1. Right-click the blue circle to the left of the line that contains the unconditional breakpoint that you want to change, or double-click the breakpoint in the Breakpoints view, and select Configure Breakpoint in the dialog.
   The Update Line Breakpoint dialog opens.

2. Select Conditional breakpoint under Breakpoint Type.
   The other fields become active.

3. Click the Variable List button, and select TASKNUM from the Variable List.

4. Click the Literals button and change the literal value to the new value where you want the breakpoint.
   The following illustration displays the finished dialog showing a conditional variable at TASKNUM=x'00003c'.



CUI--Update a Line Bpt COBOL-2--SCR

5. Click Finish.
   The dialog closes and the breakpoint displays in the Breakpoints view as a conditional breakpoint.

6. Repeat steps 1 through 5, and make statements 480 and 482 unconditional breakpoints.

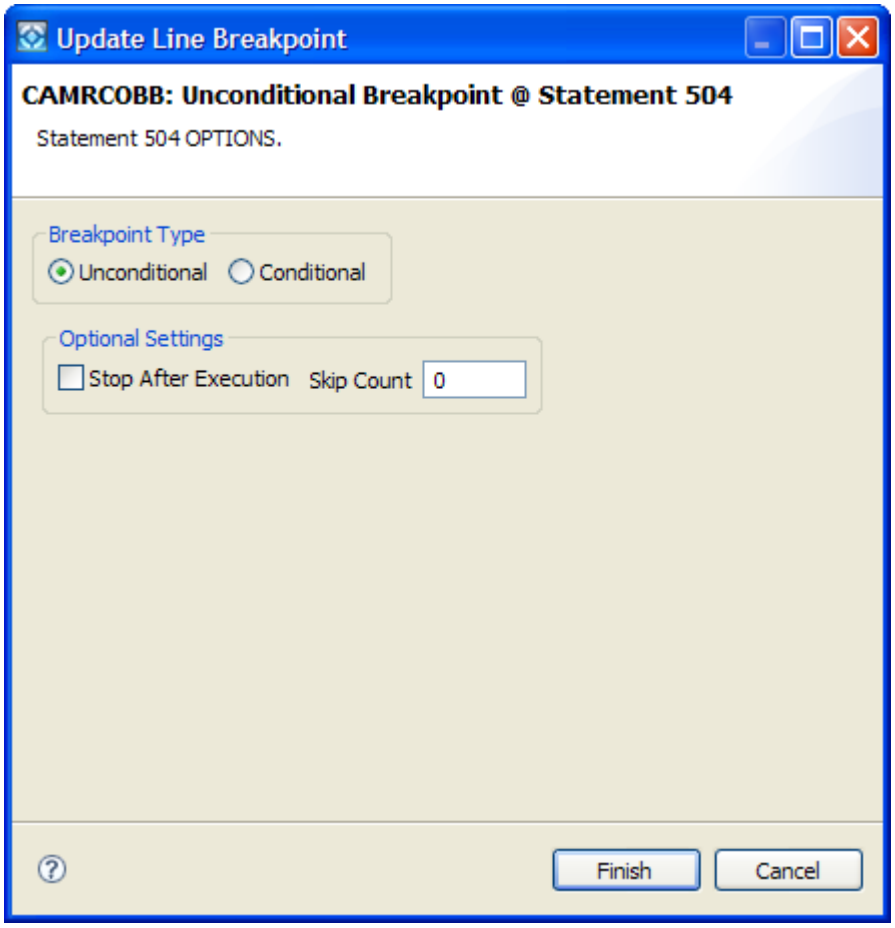
7. Change the breakpoint at statement 484 back to an unconditional breakpoint.

Change Breakpoints

The following procedure describes how to change breakpoints in batch applications.

Follow these steps:

1. Right-click the blue circle to the left of the line where you have set an unconditional breakpoint, and select Configure Breakpoint in the dialog.
   The Update Line Breakpoint dialog opens.



CUI--Update a Line Bpt COBOL Batch-1--SCR

2. Click the Conditional breakpoint option button.

3. Change the fields as follows:
   Left Side - LOOP-OUT
   Operator - GT - Greater Than
   Right Side - x'00'
   Note: Character literals do not need to be prefixed with a C. The system automatically recognizes them as characters. Thus, 2 is treated as F2, whereas hex literals must be prefixed with an X.

4. Click Finish.
   The dialog closes and the breakpoint displays in the Breakpoints view as a conditional breakpoint.

How to Delete Breakpoints

As you continue testing and debugging it is good practice to delete breakpoints you no longer need, so that the program will not stop unnecessarily.

It is easier to delete individual or multiple breakpoints from the Breakpoints view when you have a number set throughout a large program and you do not want to search through the source listing for them.

You can delete breakpoints in the following ways.

Hover the mouse on the vertical ruler to the left of the statement whose breakpoint you want to delete, right-click, and select Delete Breakpoint. The blue circle disappears, and there is no longer a breakpoint at this line.

- Select the breakpoint from the list of breakpoints in the Breakpoints view, right-click, and select Delete Breakpoint. You can also use the <Del> (Delete) key. Do not delete the unconditional breakpoint set at statement 484.

You can use the Eclipse UI delete several breakpoints at one time, follow these steps:

Click the first breakpoint on the Breakpoints view, hold down the Shift key, then click the last breakpoint you want to delete.

The breakpoints you select are highlighted.

- Right-click and select Delete Breakpoint.
  The breakpoints you select are deleted from the Breakpoints view and the program listing.

Bookmarks

Contents

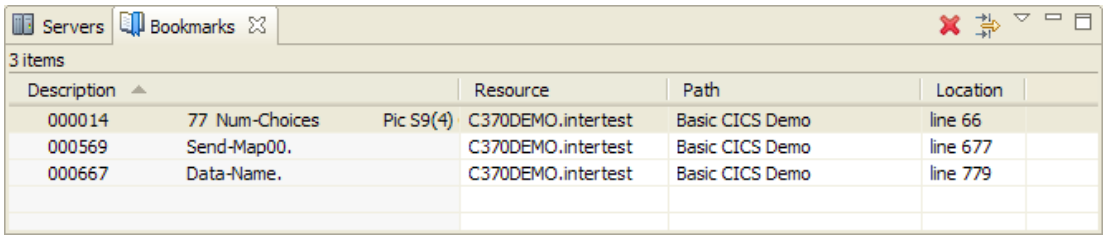
- Add Bookmarks (see page 39)
- How to Remove Bookmarks (see page 39)

Bookmarks help you to navigate quickly in the source code by marking individual locations. You can add and delete bookmarks as needed.

The Bookmarks view displays all bookmarks placed on a specific line of the program listing. The bookmarks appear in a table format that provides a description of each bookmark, the name of the program where the bookmarks are located, folder (path), and line number (location) in the Program Listing area.

Example

The following illustration displays a sample Bookmarks view, showing three bookmarks in the C370DEMO program:



CUI--Bookmarks View COBOL CICS--SCR

Note: This view is not displayed unless you open the view by selecting Window, Show Views, Bookmarks on the CA InterTest main menu.

Add Bookmarks

You can add bookmarks to your program to help you navigate quickly through the program.

Follow these steps:

1. Double-click the demo program on the InterTest Debug view.
   The program listing displays in the Program Listing area.

2. Select the statement where you want to insert the bookmark.
   The listing is positioned at the line which includes the selected item.

3. Right-click the shaded vertical ruler on the left side of the listing view, and select Add Bookmark.
   The Add Bookmark dialog opens, with the contents of the listing statement displayed in the Enter Bookmark Name field.

4. Edit the name (description) for the bookmark and click OK.
   A new bookmark is set at the selected statement. This is indicated by a small rectangle appearing next to the scroll bar. The Bookmarks view displays the details of the new bookmark.

In the Bookmarks view, after you add a bookmark, you can double-click the bookmark to open the program listing, if not already opened. The listing is positioned at the statement specified in the bookmark and highlights the statement.

How to Remove Bookmarks

Once you have created and used bookmarks, it is a good idea to delete them, so you do not clutter up your program or the Bookmarks view.

Select the line of code that contains a bookmark, and perform one of the following steps:

- From the program listing left-hand ruler, right-click your mouse, and select Remove Bookmark from the pop-up menu.

- From the Bookmarks view, highlight the bookmark and click the Delete button (the red X) on the Bookmarks view toolbar, or use the Delete key on your keyboard.

The bookmark is removed from the Bookmarks view and the program.

Visual Debugger

The Visual Debugger is a visual representation of a running CICS COBOL transaction or COBOL batch program. View the trace data that is represented by the graph structure to become better oriented with the execution flow.

Getting Started with Visual Debugger
Displays instructions on how to add the visual debugger view and how to configure CICS COBOL or batch programs for visualization.

Visual Debugger Key Features:

- **Statement Trace Visualization**
  See the transfers of the control between monitored programs and the paragraphs.
- **Dynamic Graph Update**
  Proceed through the breakpoints and view dynamic updates to the graph.
- **Navigate Execution Flow History**
  Navigate through the history of the Execution Flow.
- **Link with Editor**
  Use the graph elements to easily locate the corresponding programs, paragraphs, and control transfers in the COBOL source code.
- **Zoom and Pan**
  Zoom and pan the graph to focus on an area of interest.
- **Save and Load Session Data**
  Save debugging session data and analyze it later or forward it by email.

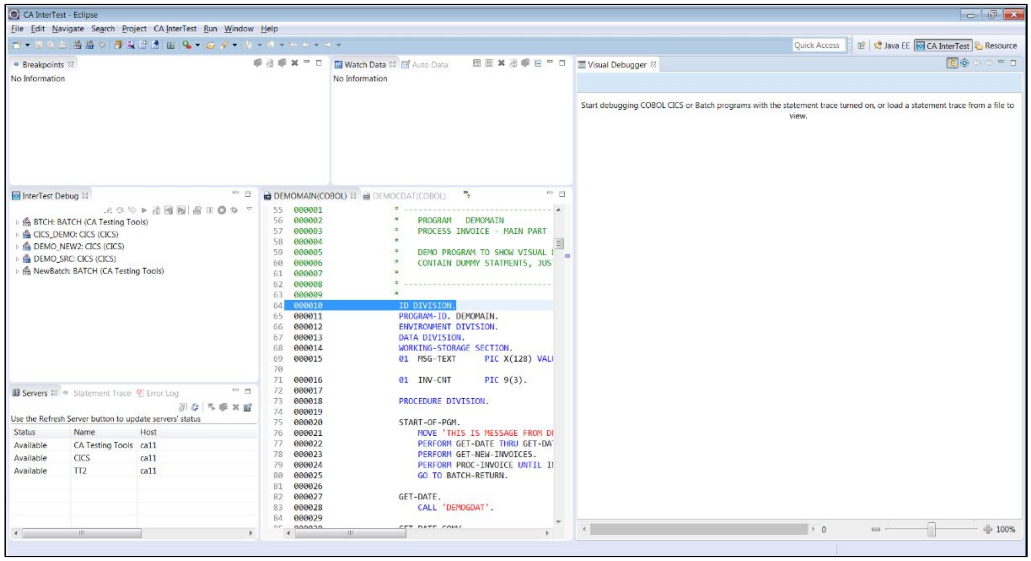
Getting Started with Visual Debugger

The latest version of the Eclipse plug-in is required to run the CA InterTest Visual Debugger. The Visual Debugger view is the default part of CA InterTest perspective.

If you have hidden the Visual Debbuger view and want to show again, go to **Window** | **Show View** | **Other**.

To customize your visualization

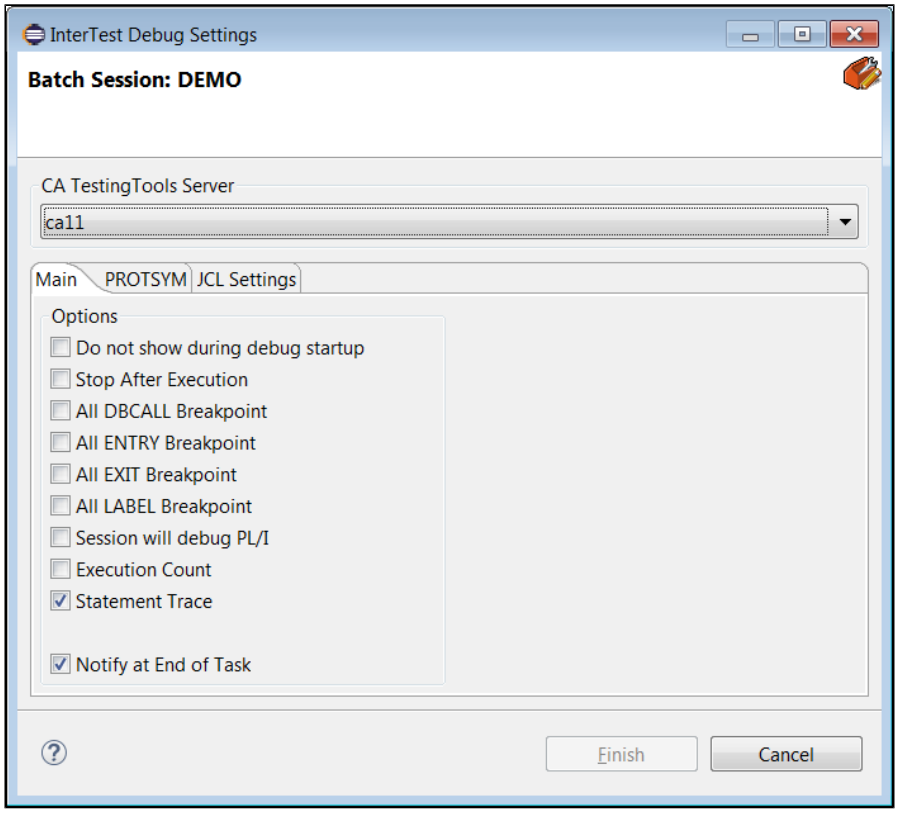
1. Expand the CA InterTest node and choose Visual Debugger.
2. Click OK.
3. Adjust the size and the position of a new view according to your preferences.



empty debugger screen

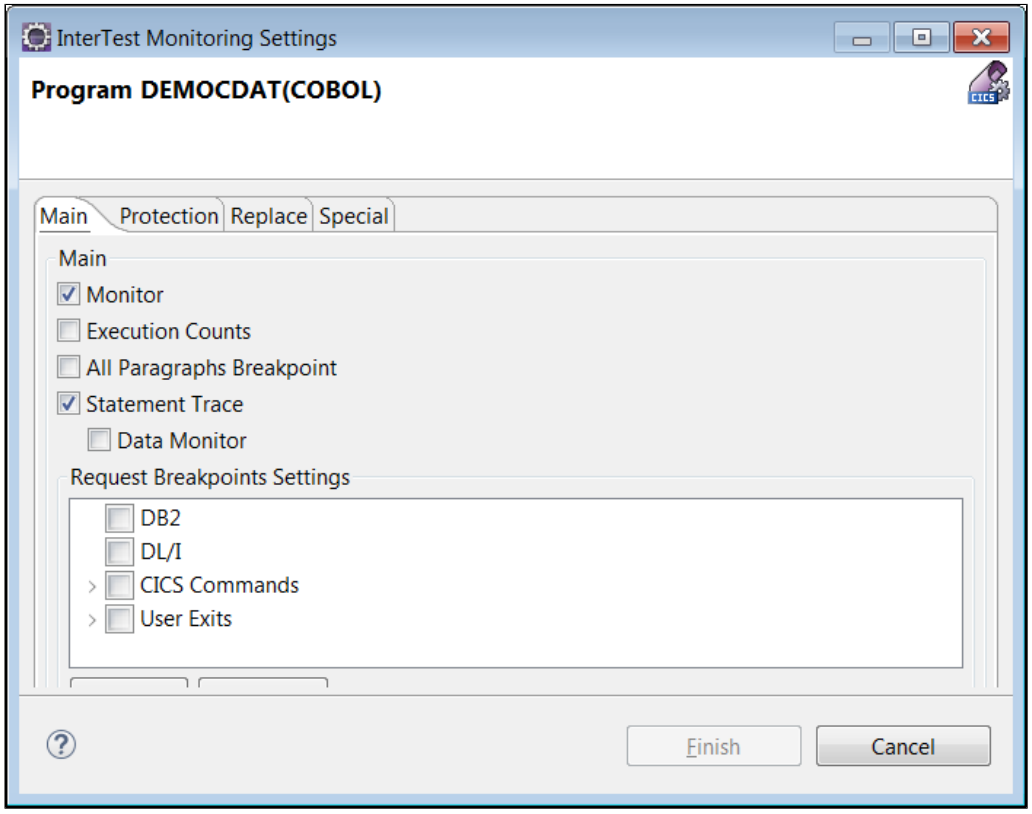
4. Enable the 'statement trace' option in order to get the data that you need for the visualization during debugging.

   Batch project:
   Ensure that the **Statement Trace** and **Notify at End of Task** options are checked in the debug settings.



Batch Session Screen View

CICS project:
Ensure that the corresponding checkboxes are checked in the program settings of each monitored program.



Debugger Program DemoCDAT Screen View

5. Submit a job for debugging (Batch project) or connect to a CICS region and run a CICS transaction.
   The graph appears once you start getting statement trace information.

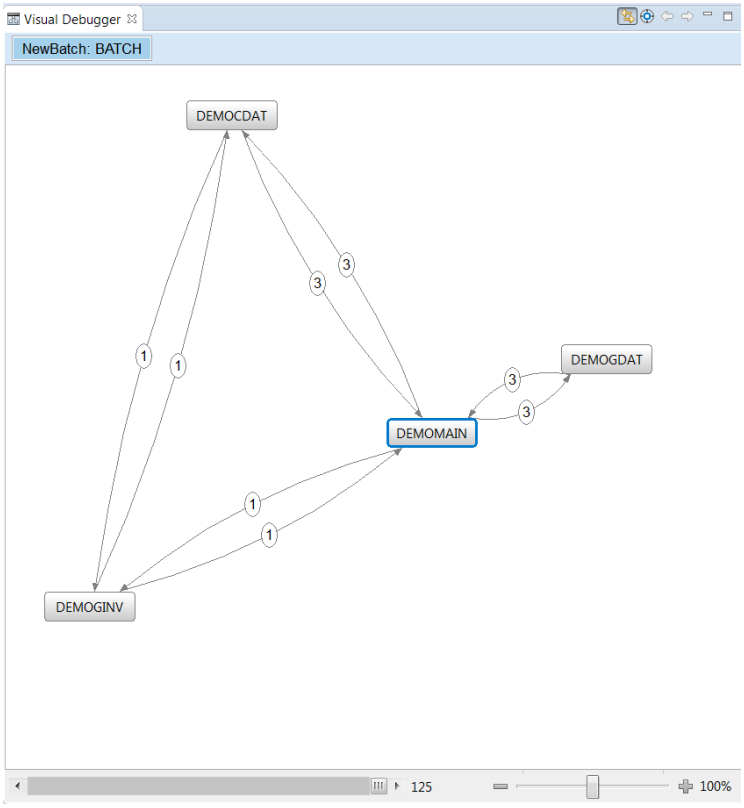
   The graph is dynamically updated until you hit a breakpoint or the execution finishes.

Statement Trace Visualization

The visual debugger graph represents a traced execution flow. There are two visualization levels:

- Programs Level
- Paragraphs Level

Programs Level



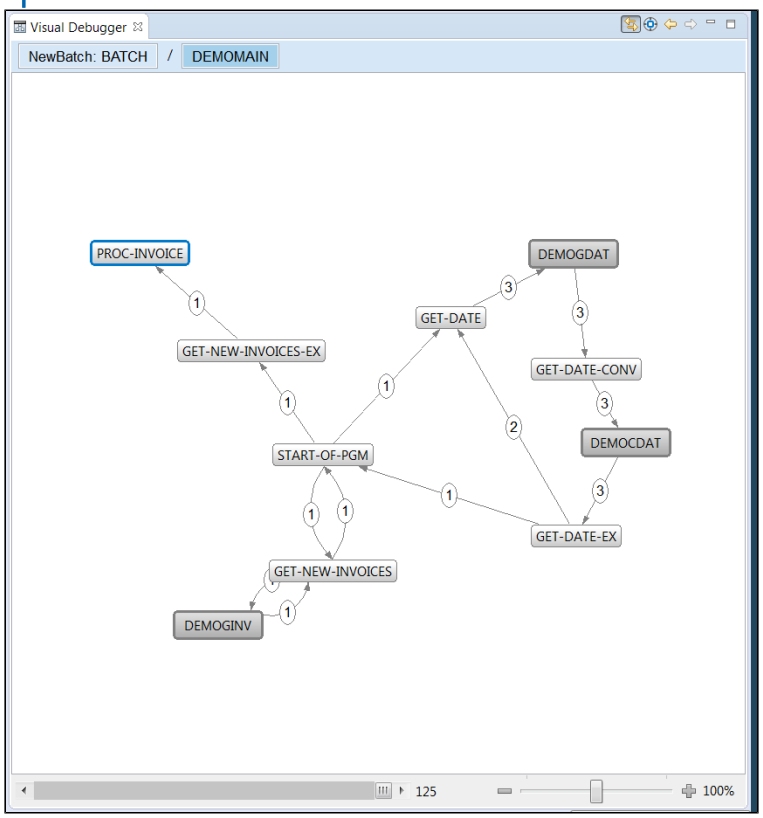
On the programs level, each node represents a monitored program that was or is currently being executed. The executed program is marked with a blue border.

The arrows represent transfers of controls between the programs. The number on an arrow shows how many times the control was passed from the program (represented by the arrow source node) to the program (represented by the arrow target node).

Right click on an arrow to access the context menu that displays up to 3 of the last transfers. Click the context menu item that represents the transfer to position your cursor in the editor of the statement that immediately precedes the transfer.

Double click any program node to switch the paragraphs level. You are also able to switch to the paragraphs level from the context menu.

Paragraphs Level



The paragraph level nodes represent the executed paragraphs or the paragraphs in the middle of execution of the program. It also visualizes other monitored programs the control was transferred to or from during the execution. The programs nodes have a darker background that allows for easier identification.

For example, **DEMOGDAT** is a program and **GET-DATE** is a paragraph.

Navigation

You can use the breadcrumb navigation on the top of the Visual Debugger view .

Both at program and paragraph levels you can find the node representing currently executed programs or paragraphs.

Click the Focus on Current Statement icon to locate the node.

Navigate between the graphs using the arrows.

Cursor Arrows Icon

Visualization of an Entry Point

The node that represents the first program executed in the debugging session is marked with a flag.

When the first executed program is visualized on the paragraphs level, the first paragraph executed in the debugging session is also marked with the flag.

Dynamic Graph Update

The graph is updated dynamically when the new debugging data is received. When you hit a breakpoint, you always receive new data.

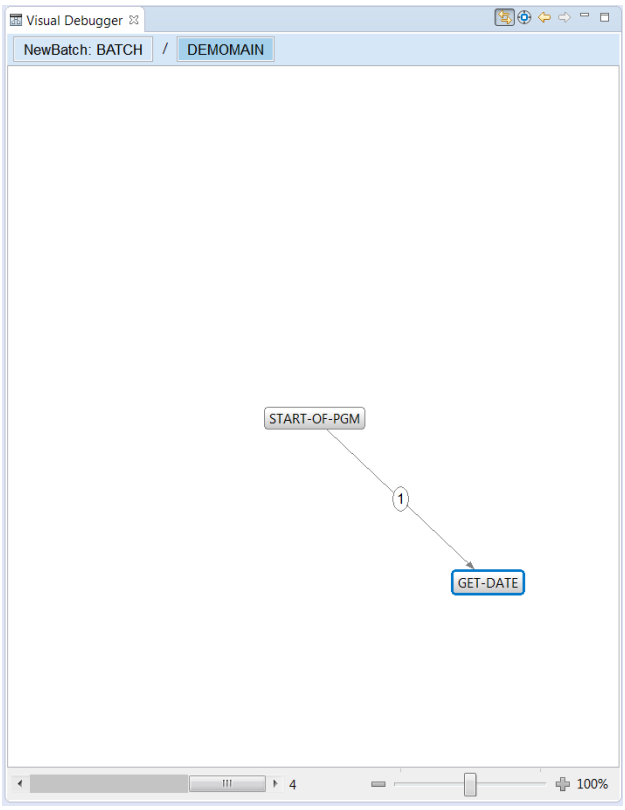
The layout is automatically adjusted during the addition of a new node or arrow.

Example: Place two breakpoints at the statements 000028 and 000034 in the DEMOMAIN program:

000020 START-OF-PGM.
000021 MOVE 'THIS IS MESSAGE FROM DEMOMAIN' TO MSG-TEXT.
000022 PERFORM GET-DATE THRU GET-DATE-EX 3 TIMES.
000023 PERFORM GET-NEW-INVOICES.
000024 PERFORM PROC-INVOICE UNTIL INV-CNT = 0.
000025 GO TO BATCH-RETURN.
000026
000027 GET-DATE.
000028 CALL 'DEMOGDAT'.
000029
000030 GET-DATE-CONV.
000031 CALL 'DEMOCDAT'.
000032
000033 GET-DATE-EX.
000034 EXIT.

Example Image Breakpoint at 000028:

The Visual Debugger view appears when the execution stops at the first breakpoint:

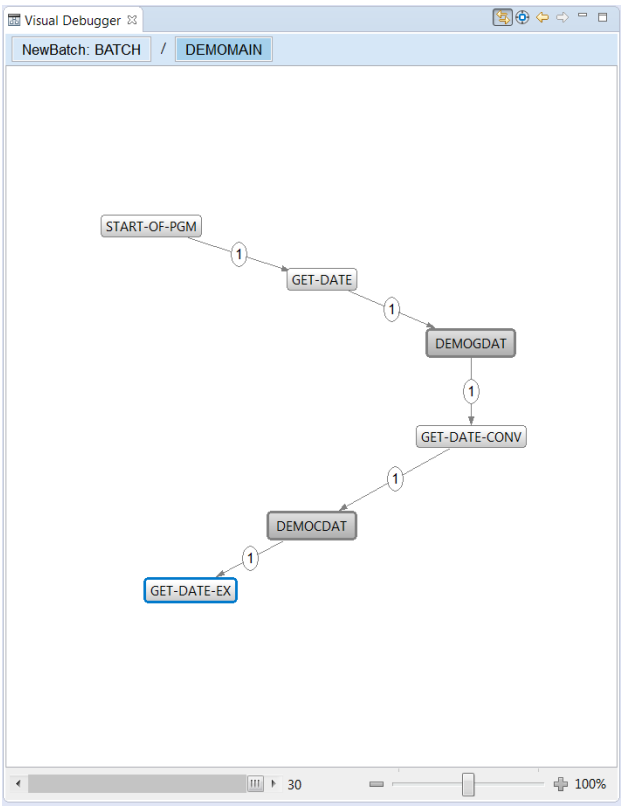


Breakpoint at 000028

Note: The **GET-DATE** paragraph is shown because statement **000028 CALL 'DEMOGDAT'** belongs to this specific paragraph.

Example Image Breakpoint at 000034:

At the second breakpoint, the graph view is updated.



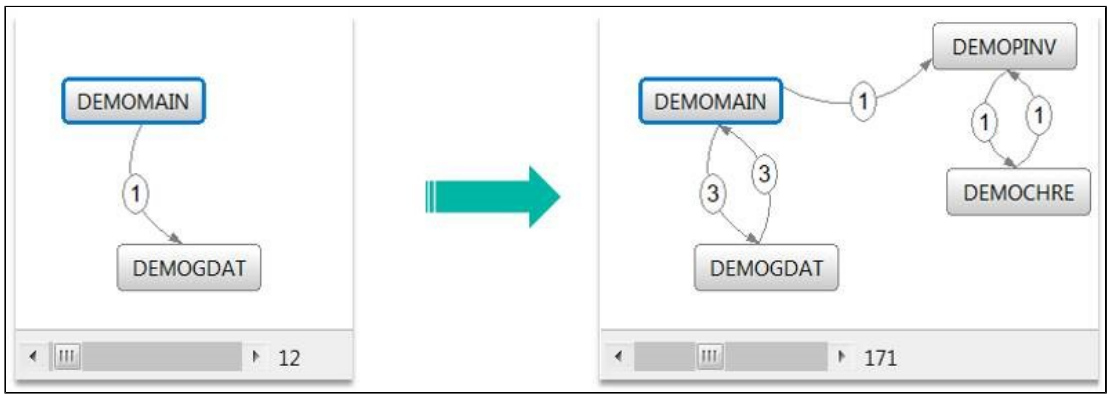
Breakpoint at 000034

Note: Generally the positions of the nodes are not determined and can differ every time the graph is redrawn.

Navigate Execution Flow History

Navigate through the execution flow history to analyze code behavior with ease.

The footer of the graph view displays a slider with the total number of statements executed since the debugging session was started. This number is located in between the history slider and the zoom slide bar.



History slider and zoom slider

Click and drag the slider back along the timeline to control which part of the statement trace is visualized. This functionality allows you to play back every event that has taken place since the beginning of the debugging session. This playback is useful if you want to see what was executed after a specific statement, step-by-step, without time-consuming inspection of the statement trace.

The nodes that represent the latest statements disappear in the flow the further you go back in history.

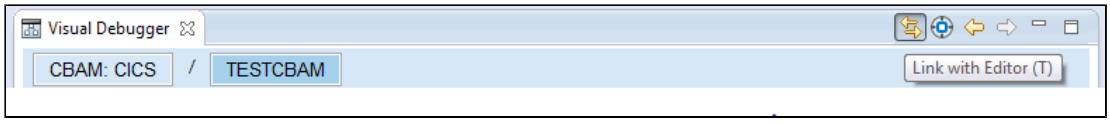
Additionally, the history slider affects:

- The numbers on the arrows reflect how many times a transition of control occur between two corresponding programs or paragraphs.
- The context menu that appears when you right-click on an arrow, and shows up to three of the most recent transfers of control.
- Annotations which appear in the editor when you click on an arrow and mark all the statements after the transfer of control occurs.

Link with Editor

Link with Editor is the feature that allows you to link Visual Debugger graph elements with the source editor. Link with Editor enables the usage of Visual Debugger graph elements for switching between the code sections. Link with Editor functionality works in both directions and allows to position the graph based on where the cursor is placed in the editor.

By default, the Link with Editor feature is turned on. To toggle it, you can use the corresponding toolbar button in the view.

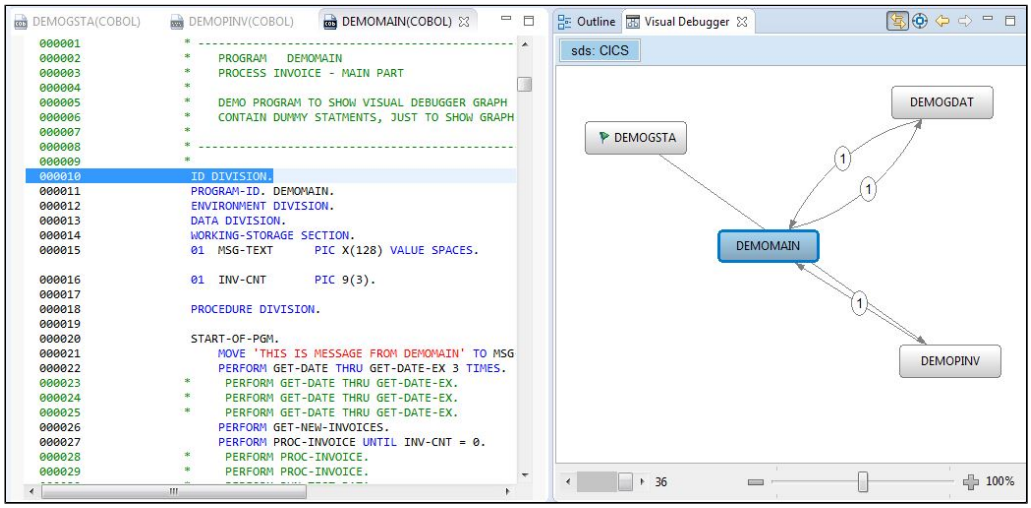


Link with Editor Icon Tool Bar

Enabling Link with Editor leads to the following behavior.

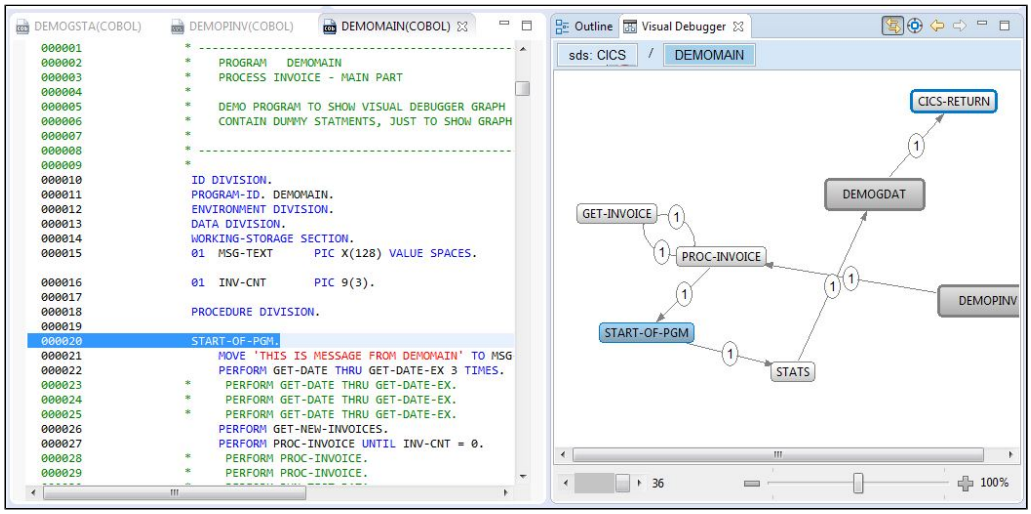
Navigate in the Editor using Graph Elements

1. Selecting a node that represents a program positions the cursor to the beginning of that program. The corresponding tab in the editor automatically opens and becomes active.



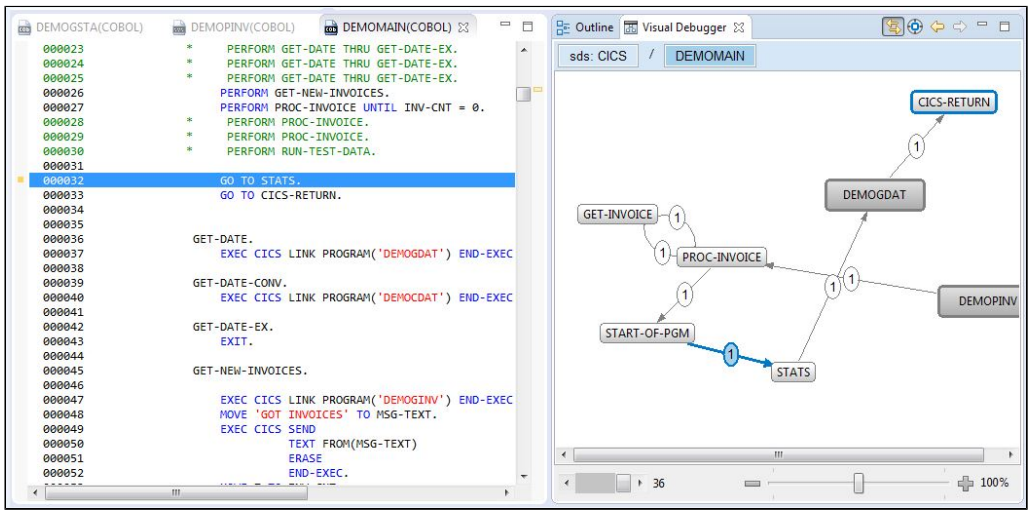
Selecting Node in Debugger

2. Selecting a node that represents a paragraph positions the cursor to the beginning of that paragraph. The tab in the editor, which corresponds to the program with that paragraph, automatically opens and becomes active.



Node in Paragraph Position

3. Selecting an edge between programs A and B positions the cursor to the line in the program A, where the last transition of the control to the program B occurred. This also applies to paragraphs level.

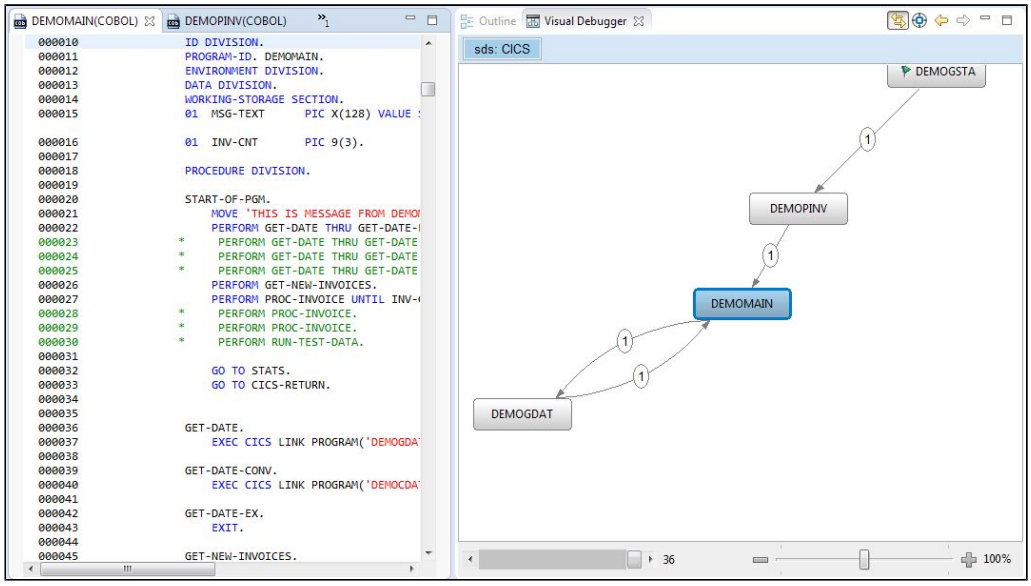


Selecting Edge Between Programs

Navigate through the Graph using the Cursor

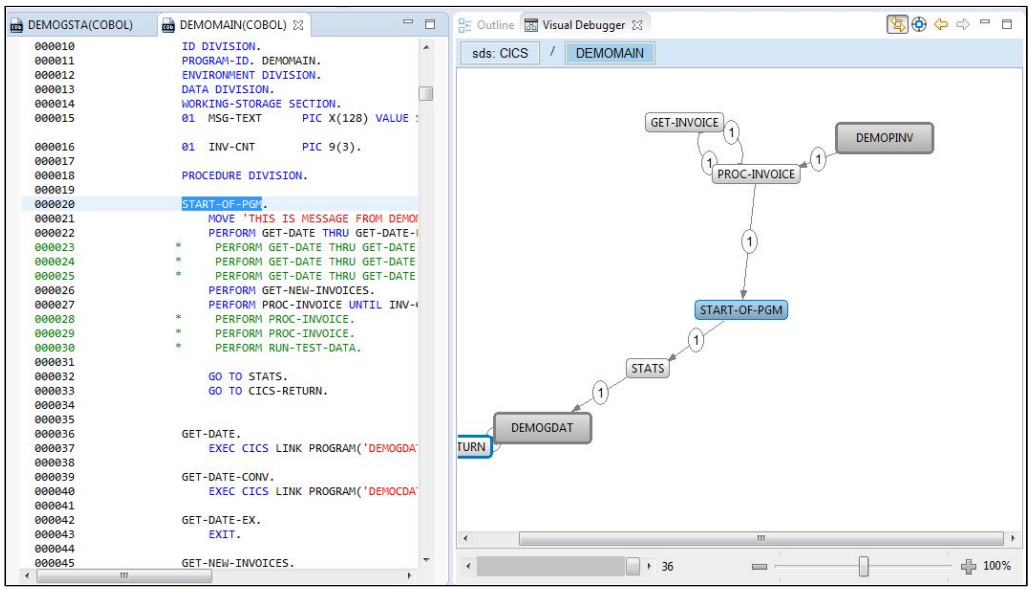
1. Placing the cursor on any line will highlight the corresponding node (program or paragraph, depending on the current level of the graph) and position it at the center of the Visual Debugger view.

Programs Level:



Debugger Actions in Program

Paragraphs Level:

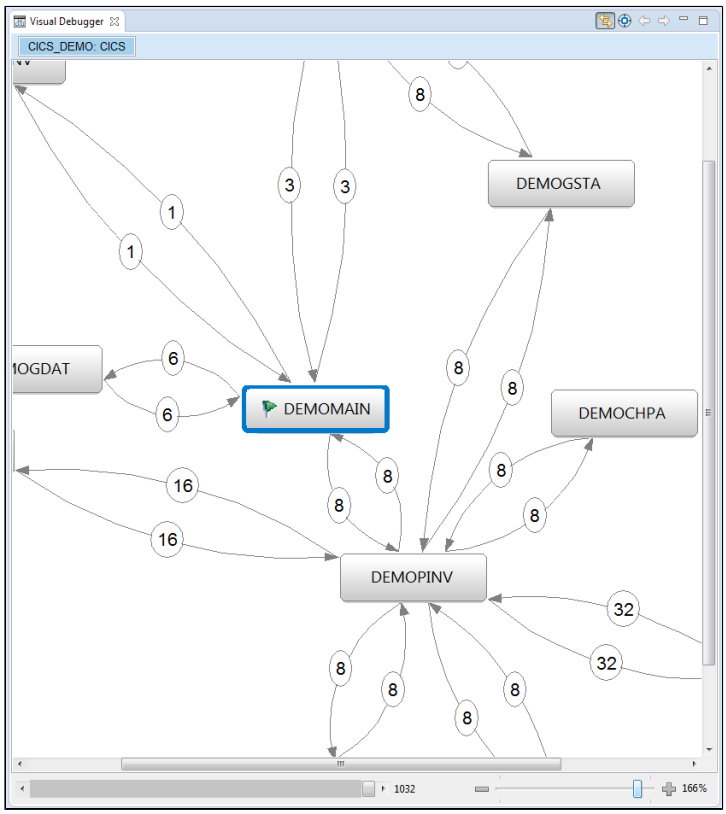


Debugger Actions in Paragraph

2. Clicking on the Position at the breakpoint will have the same effect as manual placing of the cursor on the corresponding line.

Zoom and Pan

In order to better view a graph that expands across the given user interface space, you can either click and drag desired nodes away from overlapping clusters, or zoom in on a specific area. You can pan the graph within the view.



zoom and pan

Zoom In and Out

You can zoom in and out within the view to focus on a specific area of the graph by using the control in the bottom right corner.

Pan the Graph

If you have a larger graph that expands outside of the given view, you can use panning to manipulate the image.

Node Dragging

You can drag a specific node to provide an unobstructed view of node content.

Note: Any changes to manual node arrangement will not be saved.

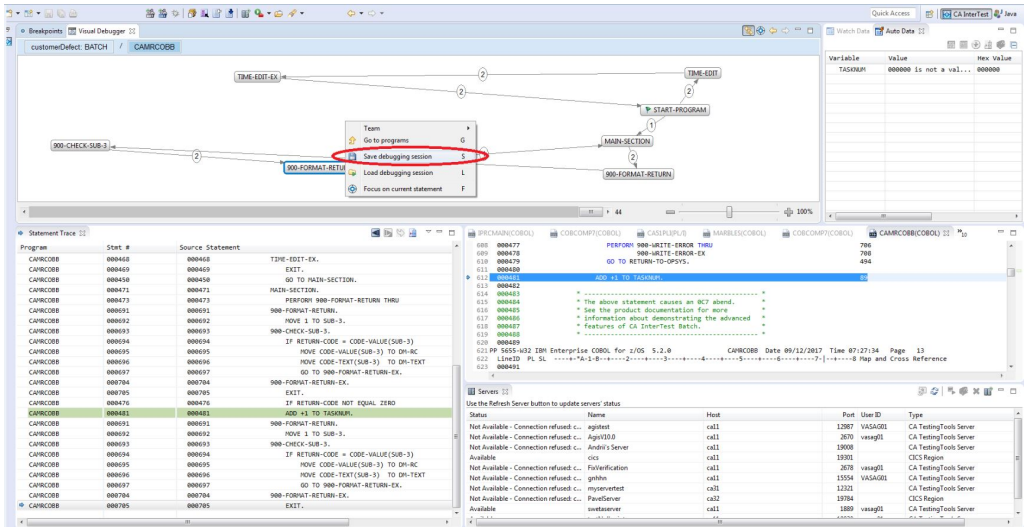
Save and Load Session Data

You are able to easily store trace data from a session that you are currently debugging into a .trace file. You can load data without the need to re-run the debugging session or connect to a mainframe.

Save a Debugging Session

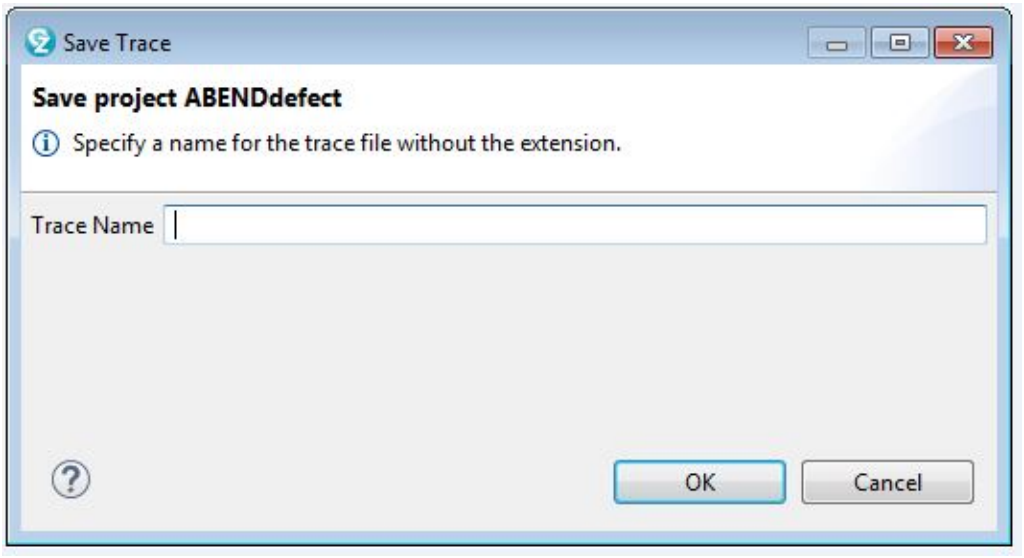
To save a debugging session, follow these steps:

1. Right click and select Save debugging session. Alternatively, you can press **S** while the Visual Debugger View is active.
   Your debugging session is saved in a .trace file inside your workspace.



Save Debugging Session Screen

The save trace dialog appears.



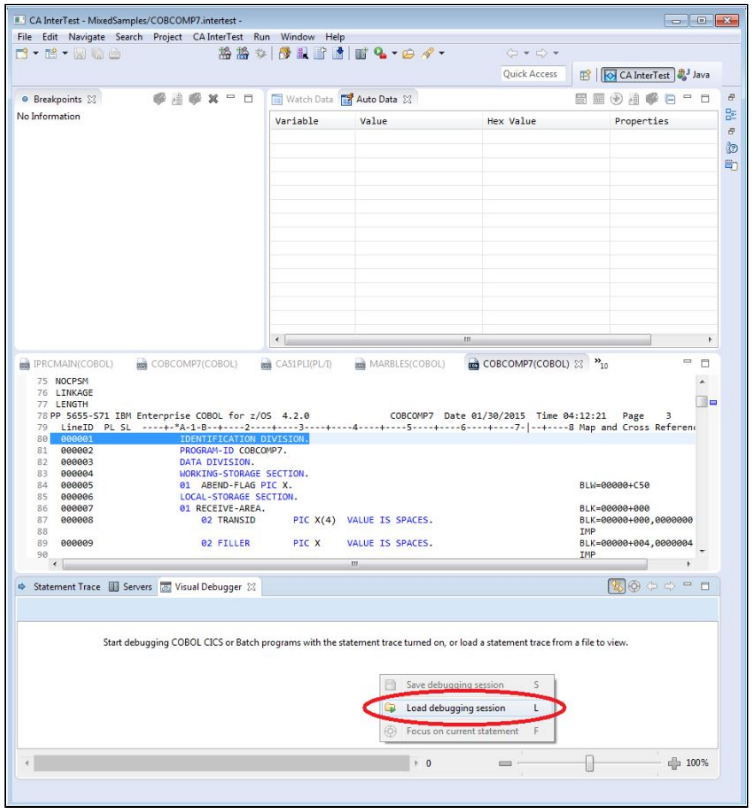
Save Trace Dialog

2. Specify the name of the trace file in the field and click OK.
   Your debugging session is saved.

Load a Debugging Session

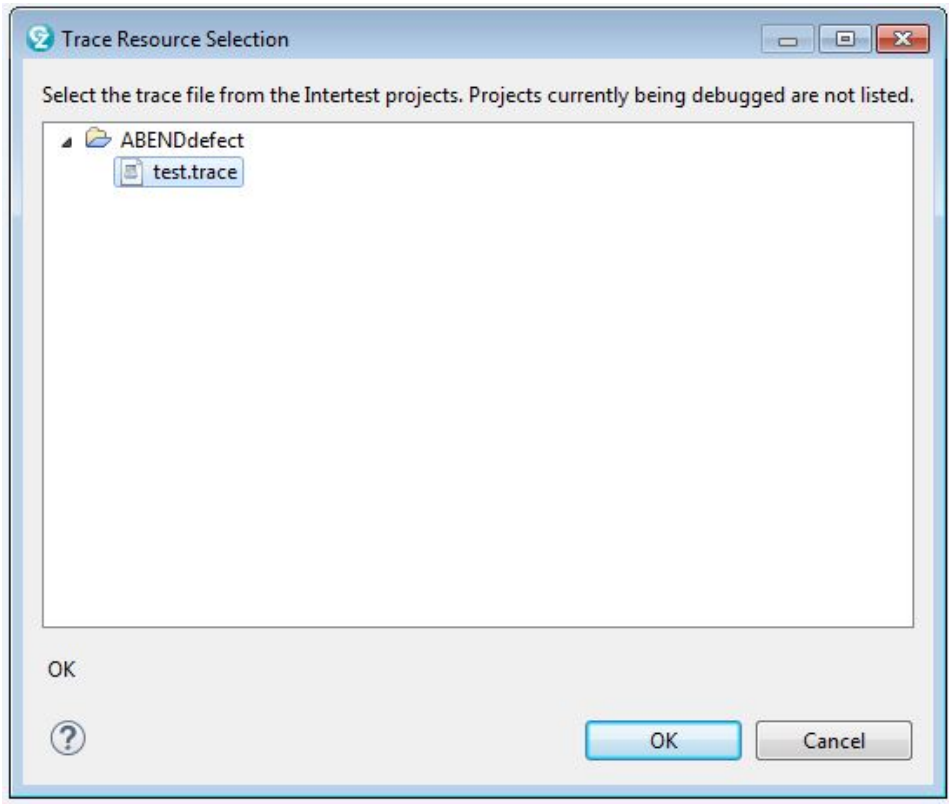
To load a debugging session, follow these steps:

1. Right click to load a debugging session. Alternatively, you can press **L** while the View is active.



Debugging Session

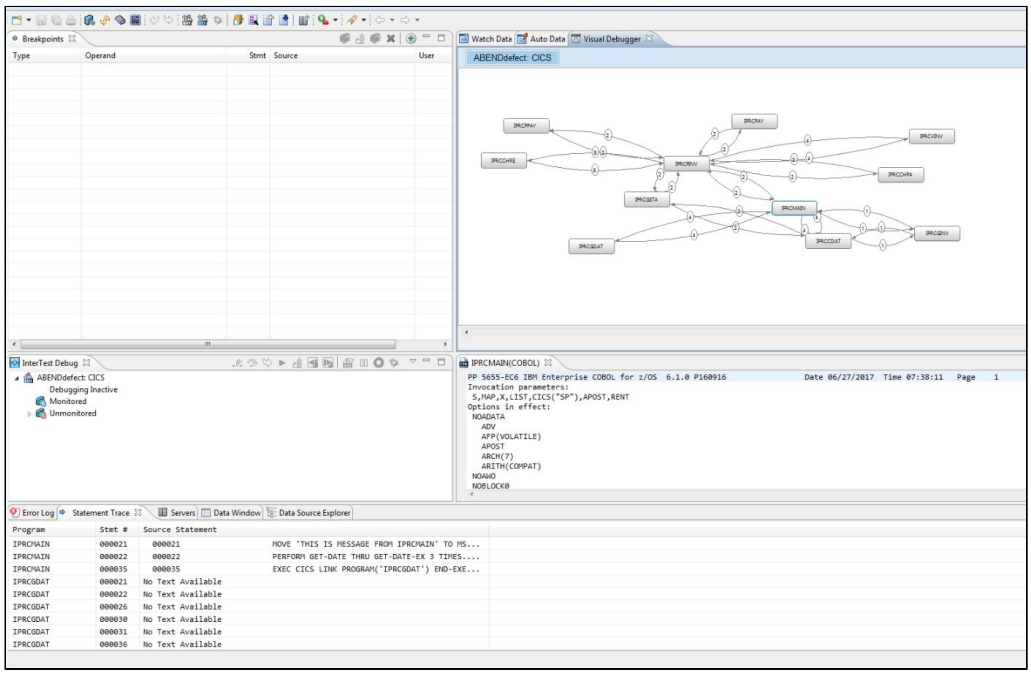
2. Select the session you want to load from the Trace Resource Selection window and click OK.



Trace Resource Selection window

Note: The projects currently being debugged are not listed.

The graph is redrawn to represent the loaded debugging session.



Redrawn Debugging Session

Note: Generally the positions of the nodes are not determined and can differ every time the graph is redrawn.