CA Technologies Product References

This document references the following CA products:

- CA InterTest™ for CICS
- CA SymDump® for CICS

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.
Contents

Chapter 1: Introduction ......................................................... 9
Getting Started with CA InterTest for CICS ......................................................... 9
Getting Help .............................................................................. 9

Chapter 2: Basic Demo Session ................................................. 11
Maintain Synchronized Processing ................................................. 11
Demo Session Objectives ................................................................ 13
Demo Session Scenario ................................................................ 13
DB2 Demo Program .................................................................. 14
Usage ....................................................................................... 14
Select the Source Program from the ITST Menus .............................. 15
View the Source Listing Profile ..................................................... 19
Set Monitoring ......................................................................... 19
View Monitoring Status ................................................................ 20
Start Program Execution ................................................................ 22
Detect and Prevent an Abend ......................................................... 23
Determine the Cause of the Error ..................................................... 24
Dynamically Change the Value in TASKNUM ..................................... 25
Control Program Execution ............................................................ 27
  How to Stop Your Program by Setting Breakpoints ................................ 27
  What You Can Do When Your Program Is Stopped ............................. 28
Set Unconditional Breakpoints ......................................................... 28
  Remove Breakpoints from the Status Display ................................... 31
Resume Program Execution ............................................................. 32
What You Have Learned ................................................................ 33

Chapter 3: Debugging Programs .................................................. 35
Checklist of Basic Debugging Tasks ................................................. 35
Display Your Source Listing .............................................................. 36
  Display Sections of Your Program ................................................. 37
  Search for Strings and Labels ....................................................... 37
Start a Test Session ..................................................................... 38
  Turn on Monitoring .................................................................. 38
Execute a Program ....................................................................... 38
Set and Remove Breakpoints ............................................................ 39
  What You Can Do at a Breakpoint ............................................... 39

Chapter 4: Defense Against Corruption .................................

Usage

Chapter 2: Basic Demo Session

1. Maintain Synchronized Processing
2. Demo Session Objectives
3. Demo Session Scenario
4. DB2 Demo Program
5. Usage
6. Select the Source Program from the ITST Menus
7. View the Source Listing Profile
8. Set Monitoring
9. View Monitoring Status
10. Start Program Execution
11. Detect and Prevent an Abend
12. Determine the Cause of the Error
13. Dynamically Change the Value in TASKNUM
14. Control Program Execution
15. How to Stop Your Program by Setting Breakpoints
16. What You Can Do When Your Program Is Stopped
17. Set Unconditional Breakpoints
18. Remove Breakpoints from the Status Display
19. Resume Program Execution
20. What You Have Learned

Chapter 3: Debugging Programs

1. Checklist of Basic Debugging Tasks
2. Display Your Source Listing
   a. Display Sections of Your Program
   b. Search for Strings and Labels
3. Start a Test Session
   a. Turn on Monitoring
4. Execute a Program
5. Set and Remove Breakpoints
   a. What You Can Do at a Breakpoint
Types of Breakpoints ................................................................................................. 39
Set Unconditional Breakpoints at Statements on the Source Listing .......................... 40
Set Unconditional Breakpoints at All References to a Data Name ............................... 42
Set Unconditional Breakpoints at Paragraph Names ................................................. 43
Inspect and Modify Main Storage .............................................................................. 45
Display the Value of a Data Item .............................................................................. 45
Modify the Value of a Data Item .............................................................................. 46
Modify the Value of a Data Item Using the MOVE Statement .................................... 46
Keeping Data Items in the Keep Window ................................................................. 47
Add a Data Item to the Keep Window (static keep element) ........................................ 48
Display the Data Item Structure .............................................................................. 49
Modify the Value of a Data Item .............................................................................. 49
Remove a Data Item from the Keep Window ............................................................. 50
Use Variable-Change Breakpoints to Detect Changing Values .................................. 50
Set a Variable-Change Breakpoint .......................................................................... 50
Remove a Variable-Change Breakpoint ..................................................................... 53
Inspect and Modify Auxiliary Storage .................................................................... 53
Inspect Auxiliary Storage ......................................................................................... 54
Modify Auxiliary Storage ......................................................................................... 55
Display Auxiliary Storage in Structured Format ....................................................... 56
Resume Program Execution ...................................................................................... 57
Resume Execution from a Specific Location ............................................................... 57
Single-Stepping .......................................................................................................... 58
Abend a Task ............................................................................................................. 59
Get Help ...................................................................................................................... 59
Get Help in Using CA InterTest for CICS ................................................................. 60
Get Help to Correct an Error .................................................................................... 61
End a Test Session ..................................................................................................... 61
Correct the Source Code ............................................................................................ 62

Chapter 4: Advanced Monitoring Features ................................................................ 63
Set Options from the Monitoring Menus ..................................................................... 63
Set and Remove Conditional Breakpoints .................................................................. 65
Remove Conditional Breakpoints from the Source Listing ........................................ 67
Remove Conditional Breakpoints from the Status Report .......................................... 68
Request Breakpoints ................................................................................................. 68
Set Request Breakpoints .......................................................................................... 69
Remove Request Breakpoints .................................................................................... 70
Backtrace Facility ........................................................................................................ 71
Access the Backtrace Summary .................................................................................. 72
Read the Backtrace Summary ..................................................................................... 72
Access the Source Listing Backtrace.................................................................73
Set the Code Counting/Coverage Option.........................................................75
Statement Trace Facility.....................................................................................76
   Enable Statement Tracing and Data Monitoring.......................................76
Navigate the Statement Trace Table...............................................................77
View Past Data Values......................................................................................78
Usage Notes ......................................................................................................78
Indirect Commands .........................................................................................79
   Code Indirect Commands.............................................................................79
How to Control the Flow of Execution............................................................82
Attach the Indirect Command to Breakpoints.................................................82
Review, Change or Delete Indirect Commands................................................84
Replacement, Protection, and Special Options...............................................84
   Replacement Options..................................................................................85
Protection Options............................................................................................85
   Special Options.........................................................................................85
Composite Module Testing .............................................................................86
Dump Analysis with CA SymDump....................................................................87
Example ...........................................................................................................88

Chapter 5: Advanced Demo Session ..................................................................89
Demo Preliminaries ........................................................................................90
   Set Unconditional Breakpoints...............................................................90
   Set Request Breakpoints..........................................................................92
Composite Support .......................................................................................94
   Check the Monitoring Status Display..................................................96
Execute the Demo Program .........................................................................97
Option 01: Replace a File Control ID ..............................................................99
   Prevent an AEIL Abend..........................................................................100
   Change the File Name............................................................................100
   Resume Execution...................................................................................101
   Review What Happened.................................................................103
Option 02: Limit CICS Storage and Requests ................................................103
   Set the MXS and MXR Options...........................................................104
   Limit CICS Requests.............................................................................105
   Remove the MXR Option.................................................................107
   Limit Acquisition of Storage.............................................................108
   Remove the MXS Option.................................................................109
   Demo Program Completes Execution...............................................109
   Review What Happened.................................................................110
Option 03: Prevent a Program from Updating a File......................................110

Contents 7
Option 04: How to Display Variable Length Data .................................................. 119
  Review What Happened ...................................................................................... 125
Option 05: How to Work with Indexed Table Items ......................................... 125
  Display an Indexed Table Entry .................................................................. 127
  Correct an Uninitialized Table Item Dynamically ...................................... 129
  Display Values of Indexes ................................................................. 129
  Review What Happened ............................................................................. 132
Option 06: How to Detect a Storage Violation .................................................. 133
  Prevent a Storage Violation ................................................................. 134
  Inspect the FREEMAN Statement ........................................................ 135
  Resume Program Execution .................................................................. 136
  Review What Happened ............................................................................. 136
Option 07: How to Test a Composite Module .................................................... 137
  Abend the Task ......................................................................................... 139
  Setting Composite Support .................................................................. 140
  Re-execute the Demo Program ............................................................. 141
  Display the Data in TASKNUM .......................................................... 142
  Correct the Data in TASKNUM ........................................................... 143
  Set a Breakpoint in Subroutine CSBIN25 ........................................... 144
  Resume Program Execution ................................................................. 145
  Remove the Breakpoint and Resume Execution ...................................... 145
  Review What Happened ............................................................................. 146

Index 147
Chapter 1: Introduction

CA InterTest for CICS is a CA product designed to simplify, improve, and shorten the testing and debugging of CICS application programs.

This section contains the following topics:
- Getting Started with CA InterTest for CICS (see page 9)
- Getting Help (see page 9)

Getting Started with CA InterTest for CICS

The main purpose of this COBOL Primer is to train new users in the basic CA InterTest for CICS features used to test and debug programs. This guide also introduces some of CA InterTest for CICS's advanced features.

All users should perform the demo sessions in this guide and become familiar with the features explained here. For complete information on these and other CA InterTest features, see the User Guide.

Perform the Basic Demo Session first, because it is the best way to begin learning about CA InterTest for CICS.

Getting Help

The Help facility provides online documentation of CA InterTest features. It makes it easy to learn and to use CA InterTest for CICS.
Chapter 2: Basic Demo Session

This chapter takes you step-by-step through the basic CA InterTest for CICS demo session. Performing the demo at a terminal is the best way to begin learning about CA InterTest. If a terminal is not available, you can read this chapter as a paper demo.

The basic demo session illustrates many of the testing and debugging tasks you will use on your own programs. These tasks are discussed in more detail in the next chapter. For more information on any topic discussed in this book, see the User Guide.

We are assuming that CA InterTest for CICS has been installed on your system. If not, contact your systems programmer.

This section contains the following topics:

- Maintain Synchronized Processing (see page 11)
- Demo Session Objectives (see page 13)
- Demo Session Scenario (see page 13)
- DB2 Demo Program (see page 14)
- Select the Source Program from the ITST Menus (see page 15)
- View the Source Listing Profile (see page 19)
- Set Monitoring (see page 19)
- View Monitoring Status (see page 20)
- Start Program Execution (see page 22)
- Detect and Prevent an Abend (see page 23)
- Determine the Cause of the Error (see page 24)
- Dynamically Change the Value in TASKNUM (see page 25)
- Control Program Execution (see page 27)
- Set Unconditional Breakpoints (see page 28)
- What You Have Learned (see page 33)

Maintain Synchronized Processing

Before you begin testing, you should be aware that when CA InterTest searches the symbolic files for the COBOL programs that you specify for testing, it tries to match the date and time in the symbolic file to that of the load module. If a match cannot be found, one of the following kinds of messages displays:

- A symbolic version list
- A warning message

This feature lets you maintain synchronized processing at all times by letting you select the correct symbolic version of the program that you want to test.

Note: For a complete description of this feature, see the User Guide.
Symbolic Version Processing

When you request a COBOL program that has no previously declared breakpoints or monitoring options set, CA InterTest matches the most recently compiled symbolic file date/time to the load module date/time.

- If an exact match is found, CA InterTest automatically selects and displays the program. However, if an exact match is found and a more recently compiled version of the program is in a PROTSYM file, CA InterTest displays the symbolic version list from which you can do the following:
  - Select the matching listing and debug the program.
  - Do not select the matching listing and cancel monitoring. Then new copy the latest version of the program into your CICS region and select the latest program version for monitoring.

- If a match is not found, CA InterTest displays the symbolic version list from which you can:
  - Select the appropriate symbolic file
  - Ignore symbolic processing for the program

The Symbolic Version List screen also explains the cause of the mismatch.

During automatic breakpoint processing, if a program has no previously selected symbolic file, CA InterTest matches the load module's date/time to a symbolic file.

- If an exact match is found, CA InterTest automatically selects and displays the program.
- If a match is not found, CA InterTest automatically selects the first symbolic file that contains the program and displays it with a warning message on the first Source Listing Breakpoint screen indicating the mismatch.

The following figure shows the Symbolic Version List screen:
**Demo Session Objectives**

When you finish the demo session, you will know how to do the following:

- Select a source listing for a program from the ITST menus
- Inform CA InterTest that you want to test a program (set monitoring)
- Respond to the information provided when a program error is detected
- Examine the value of a data item
- Dynamically change the value of a data item
- Halt program execution at any point (set breakpoints)
- Resume program execution

**Demo Session Scenario**

The CA InterTest basic demo session takes you through the following scenario:

1. You use the ITST menus to select and view the source listing of the sample program to be executed. A sample IBM Enterprise COBOL program named COBDEMO is provided for you to use.
2. You set monitoring for the sample program and view the status display of your monitoring request.
3. You exit ITST menus and initiate COBDEMO by specifying the DEMC transaction.
4. CA InterTest for CICS intercepts and prevents an ASRA abend in COBDEMO. CA InterTest halts COBDEMO at an ADD statement and displays a diagnostic message informing you that the problem was caused by improperly formatted data.
5. You examine the current value of TASKNUM, the receiving field in the ADD statement which you suspect may be the cause of the problem. You find that TASKNUM does not contain a valid packed decimal number.
6. You move a packed decimal zero into TASKNUM to correct the error.
7. You set a breakpoint to halt COBDEMO at another statement. This step is purely instructional—it has nothing to do with correcting the ASRA.

**Note:** Once a symbolic file has been selected for a program, CA InterTest continues to use the selected file and bypasses subsequent date/time matching until all declared breakpoints and monitoring options are removed for the program, or until a CNTL=NEW, PROG=program command is executed.
8. You resume program execution. CA InterTest halts the program at the breakpoint you set in Step 7. You can see that when the ADD statement executed, the value of TASKNUM changed.

9. You clean up by removing the breakpoint you set. You then resume program execution and COBDEMO runs to completion.

**DB2 Demo Program**

DB2DEMO is a sample program that helps you to set up and verify your DB2 support for CA InterTest for CICS and CA SymDump for CICS. The demo program is written in COBOL and uses basic read-only SQL statements. The table used for queries is SYSIBM.SYSTABLES, thus, no particular database for this demo is needed. The DB2DEMO program and DEMD transaction can be used in place of the COBDEMO program and DEMC transaction as described in the Demo Session Scenario (see page 13) section.

**Usage**

**Follow these steps:**

1. Make sure that you have successfully completed all of the steps described in Install DB2 Support section in CA InterTest for CICS Installation Guide or CA SymDump for CICS Installation Guide in order to support DB2. Make sure DB2DEMO was properly installed.

2. Set up monitoring:
   - To use CA InterTest for CICS to monitor the DB2 demo program, make sure InterTest is started and DB2DEMO program is monitored by it.
   - To use CA SymDump for CICS to capture the dump of the DB2 demo program, make sure SymDump is started. If CA InterTest for CICS is also installed, make sure it does not monitor DB2DEMO.

3. Run the DB2DEMO program by invoking the DEMD transaction.
   
   The welcome screen is displayed.

4. Do one of the following:
   - To abend your program without any SQL statement executed press PF2 on the welcome screen.
   - To continue and perform SQL select from SYSIBM.SYSTABLES, press Enter.
5. Following are possible errors after pressing Enter on the welcome screen:
   - DB2 connection error—Connection between the CICS region and DB2 subsystem is not established correctly.
     To correct this error make sure that the CICS region is connected to the proper DB2 subsystem. Then, run DEMD transaction again.
   - SQL error—Error occurs during the execution of a SQL statement.
     To resolve this error, use the information provided on the screen (SQLCA). Find the error in the appropriate DB2 documentation and do all necessary corrections. Then run DEMD transaction again.

6. See output of SELECT SQL statement from SYSIBM.SYSTABLES table.
   - Press ENTER to get the next screen.
   - Press PF3 or CLEAR to end the demo.
   - Press PF2 to abend the program.

7. Abend the program:
   - If DB2DEMO program is monitored by CA InterTest for CICS, the automatic breakpoint appears at "ADD 1 TO TASKNUM" statement.
   - If DB2DEMO program is not monitored by CA InterTest for CICS ASRA, the abend occurs and DB2DEMO program is terminated.

8. In case DB2DEMO is monitored for CA InterTest for CICS and you see a breakpoint appears at ADD 1 TO TASKNUM' statement, you can correct the abending statement.
   - Display TASKNUM variable using KEEP or DISPLAY command.
   - Edit value in TASKNUM so it is valid packed decimal number.
   - Continue program execution using GO command.
   - DB2DEMO program end screen is displayed and program ends.

Select the Source Program from the ITST Menus

Begin your session by selecting the source listing of the COBDEMO demonstration program from the ITST Primary Option Menu.

1. Sign on to CICS.
2. Type ITST on a clear screen.
3. Press Enter.

CA InterTest for CICS displays the Primary Option Menu:

<table>
<thead>
<tr>
<th>OPTION</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Source</td>
<td>Display/select program source files/listings</td>
</tr>
<tr>
<td>2 Monitoring</td>
<td>Display/modify CA InterTest monitoring/activity</td>
</tr>
<tr>
<td>3 Main storage</td>
<td>Display/modify CICS storage areas</td>
</tr>
<tr>
<td>4 Auxiliary storage</td>
<td>Display/access databases/files/queues</td>
</tr>
<tr>
<td>5 Dump analysis</td>
<td>Invoke CA SymDump CICS dump/trace capture facility</td>
</tr>
<tr>
<td>6 Product help</td>
<td>Invoke CA InterTest product help facility</td>
</tr>
<tr>
<td>7 Status/Maintenance</td>
<td>Product status and maintenance functions</td>
</tr>
<tr>
<td>8 What's new?</td>
<td>Display information about CA InterTest V9.1</td>
</tr>
<tr>
<td>X Exit</td>
<td>Terminate menu processing</td>
</tr>
</tbody>
</table>

Copyright (c) 2014 CA. All rights reserved.

Notice the PF keys have conventional assignments for ISPF-like navigation:

- PF1 Help accesses online help
- PF3 End displays the previous panel
- PF4 Return returns to the top-level menu

Now we can take a look at the source listing of the program we will be debugging.

1. Type 1 in the Option field.
2. Press Enter.
The CA InterTest Source Menu is displayed.

```
---------------- CA InterTest for CICS V9.1 SOURCE MENU ----------------
OPTION ==> 1

Select a member list type, specifying optional criteria below.

1  Source listings  - Display/select program source listings
2  Symbolic files   - Display/select program source SYMBOLIC files

Type specific or generic program/file name(s):
  (Valid mask characters are * and/or +)
    c*________ ________ ________ ________ ________ ________

. .
```

Since we know that the COBOL demo programs all start with a c, we can filter for all members that begin with that letter.

3. Type 1 in the Option field to search for program listings.

4. Tab to the first entry field for file/program names, and type c*. The asterisk is a generic or wildcard character. It indicates that anything from this point on in a file name will satisfy the search criteria.

5. Press Enter.
The Source Listing Selection screen is displayed. It lists all program Source Listings beginning with the letter c.

The top half of the display identifies the program and provides fields for entering commands, options, statement numbers and search criteria. Depending on your session defaults, it may also display PF key assignments and available options. These have been omitted from the Source Listing displays in this guide for clarity.

The bottom half contains the source listing.

Note: The statement numbers on your screen may not match those shown in the illustrations. This will not hinder you from performing the test session.
View the Source Listing Profile

1. Type profile on the Command line and press Enter (or, press PF4 Profile) to view the PF key assignments and other session options.

   CA InterTest for CICS displays the Source Listing Profile for your current session. This lists the PF keys, options, and current environment settings relating to the Source Listing display.

```
CA InterTest for CICS V9.1 - PROSYM FILE SOURCE LISTING PROFILE
COMMAND ===>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Auto prms 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status

Display window = N N (None), T (Titles), R (Registers),
    K (Keep), P (Program)
PF7/8 amount = PAGE PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing = BEFORE Stop Before or After the next verb is executed
Stepping amount = 001 The number of verbs to execute
Auto-stepping = OFF ON to activate; press PF4 to change values
Source List BKPT = ON OFF to use the detailed breakpoint display
From terminal ID = U084 Terminal ID where the program will execute
BKPT terminal ID = U084 Terminal ID to receive the breakpoint displays
User ID = .ANY User ID who will execute this program
AutoKeep Display = ON OFF to deactivate
Code Counting = OFF ON to activate Code Coverage
SDF = DATA HEX for Hexadecimal/Character Format
```

Note: Verify that the AutoKeep Display feature is activated for this demo session. If not, change it from OFF to ON and press Enter.

Set Monitoring

The next step is to instruct CA InterTest to monitor COBDEMO. As part of monitoring a CICS command-level program such as COBDEMO, CA InterTest automatically prevents all CICS abends during execution. How this works is shown later on when CA InterTest prevents COBDEMO from abending because of an ASRA.
There are a number of different ways of telling CA InterTest to monitor the program. If you are currently viewing a program's source listing, you can type one of the following:

1. Type **monitor** on the Command line and press Enter.
   or
2. Press PF5 (notice PF5 is listed as Monitor in the previous screen).
3. Press PF5 or use the monitor command to begin monitoring the demo program. You can do this from either the Source Listing Profile or Source Listing Display.

   The screen momentarily flashes as the request is processed. The Source Listing Display screen then reappears.

### View Monitoring Status

To verify that the demo program is being monitored:

1. Type **status** on the command line, as shown in the following screen:

   ```
   CA InterTest for CICS V9.1   - PROSYM FILE   SOURCE LISTING DISPLAY
   COMMAND ===> status
   Option #       Stmt #                         Margin= 01
   Search=
   Program= COBDEMO Option #       Stmt #                         Margin= 01
   Search=
   OP TS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
   6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
   PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
   7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
   -------------------------------------------------------------------------------
   YEARWINDOW(1900)
   ZWB
   000001 ID DIVISION.
   000002 PROGRAM-ID. COBDEMO.
   ```

2. Press Enter
CA InterTest for CICS displays the Monitoring Status screen. This status shows the monitoring entry for the current program only.

---

**CA InterTest for CICS V9.1   MONITORING STATUS   ----------------**

**COMMAND ===>

Type + to expand or - collapse option levels displayed below, or R to remove option(s).

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>- COBDEMO</td>
<td>Program monitor entry</td>
<td>IBMCOB 3.2</td>
</tr>
<tr>
<td>ments ANY</td>
<td>User monitoring options</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td>Symbolic listing file</td>
<td>PROTSYM</td>
</tr>
<tr>
<td>- -SLB</td>
<td>Source listing breakpoints</td>
<td>U023</td>
</tr>
</tbody>
</table>

*** End of data ***

PF1 Help       2 Refresh    3 End        4 Return     5 Collapse   6 Expand
PF7 Backward   8 Forward    9           10           11           12

**Note:** If the Status display does not display, you have not set monitoring correctly. Press PF3 to return to the Source Listing display, and then type MONITOR on the command line or press PF5 to set monitoring. Check the status again before continuing.

3. Once you have verified that the correct demo program is being monitored, press PF3 to go back to the Source Listing.

   Now exit to CICS using the fastpath entry (=x), as follows:

4. Type =x in the Source Listing Command field and press Enter. You return to the ITST menu processing.

5. From the Source Selection Menu, type =x in the Command field and press Enter to exit to CICS.
Start Program Execution

Now you are ready to begin testing COBDEMO.

1. Clear the CICS display and type `demc`, the transaction that executes COBDEMO.
2. Press Enter.

   The following Welcome screen displays.

   ```
   **************************************************************
   **************************************************************
   *****                             Welcome to the             *****
   *****                       CA InterTest Demo Session         *****
   *****                                                                    *****
   ***** Before proceeding, please have on hand the                  *****
   *****                guide which accompanies the Demo Session.       *****
   *****                                                                    *****
   ***** Please make sure that the program COBDEMO is monitored by     *****
   *****                CA InterTest. This program will abend if it is not *****
   *****       monitored.                                             *****
   ***** To turn the monitor on, press CLEAR and follow the steps       *****
   *****        outlined in the documentation.                         *****
   ***** If the monitor is already on, press ENTER to begin the        *****
   *****                Basic Demo Session or PF2 to go to the Options *****
   *****                        Menu.                                    *****
   **************************************************************
   **************************************************************
   
   3. Press Enter.

   COBDEMO resumes execution.
Detect and Prevent an Abend

The next screen you see is not displayed by the COBOL demo. It is a diagnostic screen, called a breakpoint display, which is displayed by CA InterTest when it detects an error.

Notice that CA InterTest has highlighted an ADD instruction and displayed a message below it.

Execution of that ADD instruction triggered an ASRA abend. CA InterTest prevented the abend and then displayed the diagnostic screen you are currently viewing.

When CA InterTest stops program execution, we say that it halts the program at a breakpoint. This can be done automatically, or CA InterTest halts a program at a breakpoint that was set by you, the programmer. The A to the left of the ADD statement indicates that the current halt in program execution is an Automatic breakpoint—not one set by you.

Now look at the highlighted message. It explains that the problem was caused by improperly formatted data. Which data? Since ADD +1 TO TASKNUM triggered the breakpoint, it is likely that TASKNUM contains improperly formatted data.

So far, CA InterTest has prevented COBDEMO from abending and identified the problem.
Next, perform the following steps:

1. Determine the cause of the error.
2. Dynamically correct the error by changing the value of TASKNUM.

**Determine the Cause of the Error**

Next confirm that the value stored in TASKNUM is not in valid arithmetic format by examining its current value; that is, its value prior to the execution of the ADD statement that triggered the ASRA.

CA InterTest for CICS displays the current value of TASKNUM, as shown in the following screen:

When the AutoKeep Display feature is active, CA InterTest automatically displays the value(s) of the variable(s) associated with a line of code. In this case, TASKNUM has been displayed in this specially formatted Keep window. On the left is the name of the data item; on the right is the fields’ datatype appropriate display format. The question mark in the beginning of the field indicates that it contains incorrect values for its datatype.

Up to six data items will be displayed at a time within the scrollable Keep window. This feature lets you see how the values of data items change as your program executes. The Keep window remains until you remove all data items from it.
Dynamically Change the Value in TASKNUM

Note: A full discussion of the Keep window is described in Keeping Data Items in the Keep window in the chapter "Debugging Programs (see page 35)". Now take a look at the contents of TASKNUM. It does not contain a valid packed decimal value. Instead, it contains low-values (binary zeros). COBOL does not allow you to add a value to a field without initializing it.

Dynamically Change the Value in TASKNUM

Now that we have identified and confirmed the cause of the problem, we can fix it.

1. Type in **m** (modify) to the left of TASKNUM in the Keep window, as shown in the following screen.
2. Press Enter.

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND =====>
Program= COBDEMO Option # Stmt # Margin= 01
Search=

---------- TASKNUM | ?00000.

000477 CONTINUE TASK.
000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
```

CA InterTest for CICS generates a fill-in-the-blanks MOVE statement, as shown in the following screen.

3. Type in **zeros** in the MOVE field, as shown next, and press Enter.
Note: You do not have to know the type of data (binary, packed, and so on) or the length of TASKNUM. The CA InterTest COBOL-like MOVE statement automatically takes care of that for you.

CA InterTest for CICS executes the MOVE statement. As a result, TASKNUM now contains a packed decimal zero. CA InterTest also displays a main storage display that shows the new value of TASKNUM, as shown in the following screen.

You can change the contents of a field simply by overtyping the desired bytes in the Keep window or on the main storage display. Overtyping is, of course, faster than using a CA InterTest-generated MOVE statement. However, with the MOVE statement you do not have to know the internal representation of the data. CA InterTest automatically takes care of the details for you.
Now we will return to the program listing so we can continue to test COBDEMO.


CA InterTest for CICS redisplays the breakpoint screen without the explanation of the abend. Note the new value of TASKNUM in the Keep window.

Note: You also can correct this bug using indirect commands, which are described in Using Indirect Commands in the chapter "Advanced Monitoring Features (see page 63)."

Control Program Execution

Now that the value in TASKNUM has been properly initialized, the next step might be to continue testing by resuming program execution. However, this is a good opportunity to learn about another important CA InterTest feature—the ability to control program execution by setting breakpoints.

How to Stop Your Program by Setting Breakpoints

One of the problems with traditional testing methods is that you have little or no control over program processing. You initiate the task, and the program either runs to completion or abends.

With CA InterTest, you can control program execution in a number of ways. For example, you can set stops, called breakpoints, anywhere in your program. Four types of breakpoints you can set are unconditional, conditional, variable-change, and request breakpoints.

- When you set an unconditional breakpoint at a statement, the program stops just before the statement is executed.
- When you set a conditional breakpoint at a statement, the program stops only if a condition you specified is met—such as a counter equaling or exceeding some value. You can also set a conditional breakpoint to stop at any instruction when the condition you specified is met.
- When you set a variable-change breakpoint, the program stops at any instruction if the value of the variable you specify has changed. This is a special type of conditional breakpoint.
- When you set a request breakpoint, the program stops before all CICS commands, macros and other program calls, such as calls to DL/I or DB2, or just before specific CICS commands, such as all READ or WRITE commands.
What You Can Do When Your Program Is Stopped

Once a program is stopped, you can use the CA InterTest testing and debugging facilities to do the following tasks:

- Examine the source listing
- Examine and modify main and auxiliary storage to detect and correct errors
- Set and remove breakpoints
- Examine a program’s backtrace, or execution path
- Keep data items in a Keep window 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

Note: All of these activities are described in detail in the next chapter.

Now we are going to demonstrate how easy it is to control program execution by setting an unconditional breakpoint.

Set Unconditional Breakpoints

You can set unconditional breakpoints directly on the Source Listing screen just as easily as you displayed and modified TASKNUM. Type u to the left of each statement where you want CA InterTest to halt program execution. Here is how to set a breakpoint at the following IF statement:

IF TASKNUM GREATER 2
1. Type in u to the left of the statement number on the IF TASKNUM GREATER 2 line, as shown in the following screen.

2. Press Enter.

```
CA InterTest for CICS V9.1 - PROTSYM FILE  ABEND DETECTED BREAKPOINT
COMMAND ===>
Program= COBDEMO Option #       Stmt #                             Margin= 01
Search=---------------------------------------------------------------------
       TASKNUM                         |  00000.
       ----------+
       -------+
       _000477 CONTINUE-TASK.
       _000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
       A         ADD +1 TO TASKNUM.
       _000480     IF TASKNUM = 1
       _000481         MOVE 'DMAPASR' TO MAPNAME.
       _000482     IF TASKNUM = 2
       _000483         MOVE 'DMAPSUM' TO MAPNAME.
       _000484     IF TASKNUM GREATER 2

       `.
```

3. Type the go command on the command line or press PF5 to continue execution from where the program is currently stopped (at the statement ADD +1 TO TASKNUM).

CA InterTest for CICS resumes program execution and continues until it reaches the breakpoint you just set. CA InterTest halts the program before that statement is executed and displays the screen shown next.

```
CA InterTest for CICS V9.1 - PROTSYM FILE  UNCOND BEFORE BREAKPOINT
COMMAND ===>
Program= COBDEMO Option #       Stmt #                             Margin= 01
Search=---------------------------------------------------------------------
       TASKNUM                         | +00001.
       ----------+
       -------+
       _000482     IF TASKNUM = 2
       _000483         MOVE 'DMAPSUM' TO MAPNAME.
       _U         IF TASKNUM GREATER 2
       _000485         GO TO SEND-END-MSG.

       `.
```

Notice the U to the left of the highlighted statement. It identifies the breakpoint as an Unconditional breakpoint. For an Automatic, Conditional, Variable-Change, or Request breakpoint, CA InterTest displays an A, C, V, or R, respectively.

**Note:** The value of TASKNUM is now 1 because the ADD +1 TO TASKNUM statement executed successfully.
When you are stopped at a breakpoint, you can do the following tasks:

- Scroll or search through your source listing
- Examine and modify main and auxiliary storage
- Add a data item to the Keep window using the `k` line command
- Set and remove breakpoints
- Examine the program's backtrace summary
- Abend your task (with or without a dump)
- Go around a problem by resuming program execution from another location. (Type `g` next to the instruction where you want to resume.)

When debugging your own programs, you will typically perform one or more of these activities, which are described in detail in the next chapter. However, COBDEMO does not have any more errors, so we are going to complete the demo session.

As you continue testing and debugging, you should clean up by removing any breakpoints no longer needed, so that when retesting the program it will not stop unnecessarily. Next, remove the unconditional breakpoint you just set so COBDEMO will not stop at this statement when it is re-executed.
Remove Breakpoints from the Status Display

If you want, you can also remove a breakpoint from the Status display:

1. Type `status` on the command line.
2. Press Enter.

CA InterTest for CICS displays the Monitoring Status screen:

```
---------------
CA InterTest for CICS V9.1   MONITORING STATUS  ---------------
COMMAND =>

Type + to expand or - collapse option levels displayed below, or R to remove option(s).

   Option     Description                Attributes
-----------  ---------------------------    ----------
 _ - COBDEMO Program monitor entry        IBMCOB 3.2
     | - Waiting at breakpoint Task 00040, UBP since 06:48 p.m.
     | - -ANY User monitoring options       Active
     | - - Symbolic listing file            PROTSYM
     | - -UBP Unconditional breakpoint      #484
     | - - Option ID                        80258747
     | - - From, to terminals               U002, U002
     | - - SLB Source listing breakpoints   U002
     | *** End of data ***

PF1 Help      2 Refresh      3 End      4 Return      5 Collapse      6 Expand
PF7 Backward  8 Forward      9           10           11           12
```

3. Type an `r` to the left of the unconditional breakpoint (UBP).
4. Press Enter.
   An asterisk displays next to the breakpoint, indicating that a command has been processed for it.
5. Press PF2 to refresh the screen.
   The unconditional breakpoint is no longer displayed.
6. Press PF3 to return to the Source Listing Breakpoint screen.
Resume Program Execution

1. Press PF5 to continue program execution.

   CA InterTest for CICS resumes program execution. COBDEMO displays a screen confirming that you have successfully corrected the ASRA.

2. Press Enter.

   COBDEMO displays the following screen.

   ****************************
   ****                   ****
   ****         CA InterTest Demo Session          ****
   ****                   ****
   ****************************

   You have completed the sample CA InterTest test session. As part of this session, you:
   * displayed program source code and compiler output online
   * displayed and modified main storage
   * controlled program execution

   This is just a fraction of CA InterTest's capabilities. You can also:
   * display or modify any CICS file, or DL/1, DB2, or SQL/DS database
   * set and remove many kinds of monitoring options

   Press ENTER or CLEAR to complete the termination.

This screen reminds you that we have touched on just a few of CA InterTest's powerful yet easy-to-use testing and debugging facilities.

3. To complete this part of the sample test session, press Clear or Enter to clear your screen.
What You Have Learned

This demo session has taught you the basics of using CA InterTest to test a program. What follows is a summary of what you have learned:

<table>
<thead>
<tr>
<th>Basic CA InterTest Testing Task</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a source program for display</td>
<td><strong>ITST</strong> Menu Option 1.1, then select program from list</td>
</tr>
<tr>
<td>Set monitoring for a program</td>
<td><strong>MONITOR</strong> command or PF5 from Source Listing</td>
</tr>
<tr>
<td>View a Monitoring Status display</td>
<td><strong>STATUS</strong> command or PF12 from Source Listing</td>
</tr>
<tr>
<td>Interpret the information CA InterTest provides when it detects an error</td>
<td>View the automatic breakpoint, press PF1 for help</td>
</tr>
<tr>
<td>Examine the value of a data item in a Keep window</td>
<td>View the data in the AutoKeep Display window</td>
</tr>
<tr>
<td>Modify main storage</td>
<td>Overtyped value in Keep window or</td>
</tr>
<tr>
<td></td>
<td>Type d to left of variable, place cursor under variable, press Enter for a CORE Main Storage display; on the CORE display, overtype the values and press Enter</td>
</tr>
<tr>
<td>Set an unconditional breakpoint</td>
<td><strong>Type u</strong> to left of instruction, press Enter</td>
</tr>
<tr>
<td>Remove a breakpoint from the Monitoring Status display</td>
<td>Use PF12 to view Monitoring Status; type r next to UBP, press Enter</td>
</tr>
<tr>
<td>Remove a breakpoint from the Source Listing Display</td>
<td>On Source Listing, overtype U with x, press Enter</td>
</tr>
<tr>
<td>Resume program execution</td>
<td><strong>RESUME</strong> command or PF5</td>
</tr>
</tbody>
</table>
Chapter 3: Debugging Programs

This chapter provides an overview of key CA InterTest for CICS functions. For complete information, see the User Guide and the Help facility.

This section contains the following topics:
- Checklist of Basic Debugging Tasks (see page 35)
- Display Your Source Listing (see page 36)
- Start a Test Session (see page 38)
- Set and Remove Breakpoints (see page 39)
- Inspect and Modify Main Storage (see page 45)
- Keeping Data Items in the Keep Window (see page 47)
- Use Variable-Change Breakpoints to Detect Changing Values (see page 50)
- Inspect and Modify Auxiliary Storage (see page 53)
- Resume Program Execution (see page 57)
- Abend a Task (see page 59)
- Get Help (see page 59)
- End a Test Session (see page 61)
- Correct the Source Code (see page 62)

Checklist of Basic Debugging Tasks

The following list highlights basic debugging tasks:

- Display the source listing.
- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.
When the program is stopped at a breakpoint, you can do the following procedures:

- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Keep data items in the Keep window to observe changes in their values
- Resume execution
- Abend the task

**Note:** Before you read this chapter, you should have performed the Basic Demo Session described in the previous chapter. The demo session provides a basic introduction to CA InterTest and illustrates many of the functions discussed in this chapter.

### Display Your Source Listing

You can display your source listing online at any time from CICS through the ITST transaction or the LIST=profilename transaction (where profilename is the program name). The Basic Demo explains how to display a source listing using the ITST transaction. For additional information on how to use LIST=, see the *User Guide*.

**Note:** The standard CA InterTest transaction IDs, such as LIST, ITST, CORE, and FILE, might have been changed at your site. If any of the transactions mentioned in this guide do not function as described, contact the person who installed CA InterTest.

Depending on your defaults, the options and PF keys available to you might not be displayed at the top of the Source Listing Display. If this information is not displayed, you can view it by pressing PF4 to access the Profile screen, and then setting Display window to T.
Display Sections of Your Program

The options (OPTS) available at the Source Listing screen make it easy to display any section of your COBOL program. Type the command in the command line or the number of the section you want to display in the Option # field and press Enter:

<table>
<thead>
<tr>
<th>Command/Option #</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>L .PD</td>
<td>Procedure Division</td>
</tr>
<tr>
<td>L .WS</td>
<td>Working Storage Section</td>
</tr>
<tr>
<td>L .LS</td>
<td>Linkage Section</td>
</tr>
<tr>
<td>L .DM</td>
<td>Data Division Map</td>
</tr>
<tr>
<td>L .PM</td>
<td>Clist/Pmap</td>
</tr>
<tr>
<td>L .DX</td>
<td>Data Name Cross Reference</td>
</tr>
<tr>
<td>L .PX</td>
<td>Procedure Cross Reference</td>
</tr>
<tr>
<td>L .EM</td>
<td>Error Messages</td>
</tr>
<tr>
<td>L .LC</td>
<td>Local-Storage Section</td>
</tr>
</tbody>
</table>

Search for Strings and Labels

You can also search for and display a character string by typing a FIND string NEXT/PREV command on the command line or option 9 (Search forward) or 10 (Search backward) in the Option # field and typing the character string in the Search field. For example, the following screen illustrates how you would search backward for the string `custname`.

```
CA InterTest for CICS V9.1 - PROSYM FILE  SOURCE LISTING DISPLAY
COMMAND ===> f custname prev
Program= COBDEMO  Option #       Stmt #                             Margin= 01
Search=-----------+---------------------------------------------------------------------
  000911 VSAM-REW
  000912     MOVE 'THIS IS NOT A NAME ' TO VSAM-NAME.
000913*EXEC CICS REWRITE
  000914*     DATASET(TASK-PROTCPF)
  000915*     FROM(VSAM-AREA)
  000916*     LENGTH(REC-LEN)
  000917*     END-EXEC.
  000918     Call 'DFHEI1' using by content x'0606e0000700004000f0f5f2f4
```
Start a Test Session

To start a test session, follow these steps:

- Turn on monitoring
- Execute the program

Turn on Monitoring

To test a program, you must instruct CA InterTest for CICS to monitor it. When CA InterTest for CICS monitors a program, it detects and prevents errors before they occur, including the following violations:

- All storage violations
- All CICS abends
- Any statement that would cause a program check or other abend
- All illegal or invalid instructions that would cause CICS to crash
- All wild branches
- All violations of CICS standards

The easiest way to turn on monitoring is from the program’s source listing. Perform the following steps:

1. Display the source listing as described previously. CA InterTest displays the Source Listing Display screen for the program.
2. Press PF5 to set monitoring, which remains in effect for the program until specifically removed.

Execute a Program

Once you have turned on monitoring for a program, you are ready to execute it. Perform the following actions:

1. Press PF3 to exit the Source Listing facility. If you entered Source Listing from an ITST menu, you return to that menu.
2. To exit the ITST menus, type =x in the top field and press Enter. The ITST transaction ends and you return to CICS.
   
   **Note:** Before returning to CICS, you may want to set breakpoints directly on the source listing, as described in Setting and Removing Breakpoints.
3. On a clear CICS screen, type the transaction identifier of the program you are monitoring.
Once you enter the program's transaction ID, one of the following occurs:

- Your program runs to completion. The results might not be correct.
- CA InterTest for CICS stops your program at a breakpoint—either one set by you (for more information, see Set and Remove Breakpoints (see page 39)) or one automatically triggered by CA InterTest for CICS because it detected an error.

Set and Remove Breakpoints

When you test a program, it is important to be able to halt program execution at specified locations. A halt in program execution is called a breakpoint.

What You Can Do at a Breakpoint

When your program is stopped at a breakpoint, you can do the following actions:

- Examine the source listing
- Inspect and modify main storage
- Inspect and modify auxiliary storage
- Set and remove breakpoints
- Examine the backtrace
- Keep data items in the Keep window to observe changes in their values
- Set and remove monitoring options
- Specify indirect commands
- Resume execution
- Abend the task

We have already discussed how you can examine your source listing. The other breakpoint activities are discussed in this and the next chapter.

Types of Breakpoints

There are six types of breakpoints, as shown in the following table:

<table>
<thead>
<tr>
<th>Breakpoint Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic</td>
<td>The program stops because CA InterTest detected and prevented an error.</td>
</tr>
</tbody>
</table>
Set and Remove Breakpoints

<table>
<thead>
<tr>
<th>Breakpoint Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unconditional</td>
<td>The program stops at the location you specify.</td>
</tr>
<tr>
<td>conditional</td>
<td>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.</td>
</tr>
<tr>
<td>variable-change</td>
<td>The program stops at any location if the value of a specified variable has changed.</td>
</tr>
<tr>
<td>request</td>
<td>The program stops at every CICS command or macro, or at certain CICS commands or macros, or at calls to DL/I, DB2, or software.</td>
</tr>
<tr>
<td>single-step</td>
<td>The program stops after executing one or more verbs.</td>
</tr>
</tbody>
</table>

An automatic breakpoint occurs when CA InterTest for CICS detects an error. When a program is stopped at an automatic breakpoint, you can either correct the error or go around it. You can press PF1 to find out what caused the error and how to use CA InterTest to fix it. You set all other breakpoints.

In this chapter we are going to explain how to set and remove unconditional breakpoints as well as variable-change breakpoints, because you will use these the most.

For more information on conditional and request breakpoints, see the chapter "Advanced Monitoring Features". For more information on Single-stepping, see Resuming Program Execution. The following sections explain how to set breakpoints:

- Set breakpoints at statements on the source listing
- Set breakpoints at all references to a data name
- Set breakpoints at all or selected paragraph names

**Set Unconditional Breakpoints at Statements on the Source Listing**

To set an unconditional breakpoint on the program source listing, follow these steps:

1. Type u in column 1 next to the statement where you want the breakpoint.
2. Press Enter.
The following screen shows how to set an unconditional breakpoint.

![CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY]

```
Program= COBDEMO  Option #       Stmt #                             Margin= 01
Search=
OPTS 1 Proc d  2 Work-stor 3 Link sect 4 D-map  5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help      2           3 End        4 Profile    5 Monitor    6 Menu
7 Backward  8 Forward 9 Next Wnd 10 11   12 Status
-------------------------------------------------------------------------------
 000475     IF EIBAID = DFHPF14  GO TO EXPANDED-DEMO.
 000476     GO TO SEND-FIRST-SCREEN.
 000477     CONTINUE-TASK.
 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
 000479     ADD +1 TO TASKNUM.
 000480     IF TASKNUM = 1
  .
  .
```

After you set a breakpoint, CA InterTest redisplays the screen with the U in uppercase. The U remains until you remove the breakpoint.

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

To remove an unconditional breakpoint, overtype the U with an x, or type an r next to the breakpoint's entry on the Monitoring Status display (PF12 from Source Listing). You will want to remove breakpoints no longer needed so that when retesting the program, you are not stopped unnecessarily.
Set and Remove Breakpoints

Set Unconditional Breakpoints at All References to a Data Name

You can set unconditional breakpoints at all references to a data name from the Cross Reference section.

1. To display the Dataname Cross Reference, type l.dx in the command line or 6 in the Option # field on the Source Listing screen, as shown next, and press Enter.

```
CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> l.dx
Program= COBDEMO Option # Stmt # Margin=01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PPKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status

00477 CONTINUE-TASK.
00478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
00479 ADD +1 TO TASKNUM.

CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> l.dx
Program= COBDEMO Option # Stmt # Margin=01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PPKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status

DEF DATA NAMES REFERENCES

According to the previous example, CA InterTest will set breakpoints at every statement that references data names TASK-STRUCTURE and TASK-SWITCH.

2. Now type a u to the left of the data names for which you want to set unconditional breakpoints, as shown next, and press Enter.

```
CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> l.dx
Program= COBDEMO Option # Stmt # Margin=01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PPKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status

DEF DATA NAMES REFERENCES

According to the previous example, CA InterTest will set breakpoints at every statement that references data names TASK-STRUCTURE and TASK-SWITCH.

3. To remove the breakpoints, overtype the U with X. You can remove all the breakpoints for a data name from the Cross Reference section, or you can selectively remove breakpoints at specific statements. You can also remove individual or multiple breakpoints from the Monitoring Status display. You might want to do this when you have a number of breakpoints set throughout a large program and do not want to hunt through the source listing for them.
If your source listing does not include the Cross Reference section, you can also set a breakpoint at all references to a data name from the Working-Storage section where the data name is defined.

1. To display the definition, type the data name in the Search= field and press Enter.
2. When CA InterTest displays the definition, type a \textbf{u} to the left of the statement, as shown next, and press Enter.

In this example, CA InterTest sets breakpoints at all references to TASKNUM.

3. To remove the breakpoints, overtype the \textbf{U} with \textbf{x}, or remove the breakpoints from the Monitoring Status display.

### Set Unconditional Breakpoints at Paragraph Names

**To set breakpoints at all procedures and all labels**

1. To set unconditional breakpoints at all Procedure Names and All Labels, display the Cross Reference Table using Option # 1 (or \textbf{L.DX} command), type a \textbf{U} next to the DCL heading line, and press Enter.

A message prompts you to confirm the request for setting many breakpoints by pressing PF3.

It is easy to set unconditional breakpoints at all or selected paragraph names from the Procedure Names section.
2. To display the Procedure Names, type `l px` in the command line or a 7 in the Option # field on the Source Listing screen, as shown next, and press Enter.

```
CA InterTest for CICS V9.1 - PROSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> l px
Program= COBDEMO Option #       Stmt #                             Margin= 01
Search= +
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Mmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
+---------------------------------------------------------------------+
| XREF(FULL)            |
| 000001 ID DIVISION.   |
| 000002 PROGRAM-ID. COBDEMO. |
| 000003 ENVIRONMENT DIVISION. |
| 000004 DATA DIVISION.  |
| 000005 WORKING-STORAGE SECTION. |
| 000006 77 S999-FIELD1   PIC S9(3). |
| .                    |
| .                    |
```

CA InterTest displays the Procedure Names section.

3. Type a u to the left of the paragraph names for which you want to set unconditional breakpoints. You can also type a u on the PROCEDURE NAMES line, to set breakpoints at all paragraph names, as shown in the following screen:

```
CA InterTest for CICS V9.1 - PROSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> l px
Program= COBDEMO Option #       Stmt #                             Margin= 01
Search= +
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Mmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
+---------------------------------------------------------------------+
| DEFINED PROCEDURE NAMES REFERENCES |
| 982 AFTER-REWRITE |
| 858 CICS-LOOP     PIC S9(3). |
| .                    |
| .                    |
```

4. To remove the breakpoints, overtype the U with x. You can also remove the breakpoints from the Monitoring Status display.

**Note:** A message warns you if your specification sets more than 20 breakpoints. You can then either confirm or cancel the breakpoints.
Inspect and Modify Main Storage

The ability to inspect main storage at a breakpoint means you can determine the values of data items without using a dump. It is easy to find logic errors because you can display the value of a data item where it is defined and at any point in the code where it is referenced. And, you can dynamically change its value to correct errors or test other program paths.

Remember that changes to program storage are dynamic; that is, the change affects only the current test session. To correct program errors, you must change the source code and recompile the program.

There are many ways to inspect and modify main storage. We are going to discuss one of the easiest methods of viewing and changing program storage here. Keeping Data Items in the Keep window explains another easy way to display and modify the values of data items.

Remember, you can inspect system storage at any time, independent of program monitoring and execution.

Display the Value of a Data Item

When you are stopped at a breakpoint, you can request the display of the value of a data item directly from the source listing.

1. To display its value where it is defined, type a d to the left of the statement defining it and press Enter, as shown in the following screen.

   ![Source Listing](image)

2. To display the value of a data item where it is referenced, type a d to the left of the statement referencing it, place the cursor under any character in the data item, and press Enter.
CA InterTest responds by displaying the contents of the data item in structured format, as shown next.

```
CA InterTest for CICS V9.1  -  MAIN STORAGE UTILITY  -  Termid = U087

Starting at Address =2080A138        Structure Display Format
02 TASKNUM                       | ?00000.                    |
02 TASKNUM-CHAR                  | ...                        |
02 TASK-TEXT                      |                            |
03 TASK-ID-NO                     | 000.                       |
03 FILLER                         |                            |
03 TASK-MESG                      | THIS IS A MESSAGE          |
03 FILLER                         |                            |
03 TASK-DATE                      |                            |
04 TASK-MM                        | 12                         |
04 TASK-SL1                       | /                          |
04 TASK-DD                        | 25                         |
04 TASK-SL2                       | /                          |
04 TASK-YY                        | 99                         |

------------------------------------------------------------------------------
PF1 Help       2            3 End        4 Return     5            6 Dump
PF7 Backward   8 Forward    9 Caps Off  10           11 Redisplay 12 Structure
CORE='TASK)
CAIN0452 FIELD DOES NOT CONTAIN A VALID PACKED DECIMAL (COMP-3) VALUE
```

Note: CA InterTest displays more than just the contents of the specified data item (TASKNUM); it also displays all the items below it in the same COBOL structure.

3. Press Clear or PF3 to return to the source listing.

Modify the Value of a Data Item

You can modify the value of a data item by overtyping the bytes in the main storage display and pressing Enter. For example you could change the contents of TASKNUMB in the previous figure by overtyping the question mark with a zero, CA InterTest takes care of the datatype appropriate internal conversion automatically.

Modify the Value of a Data Item Using the MOVE Statement

Overtyping the main storage display is the easiest way to change the value of a data item. However, you want to use the CA InterTest COBOL-like MOVE statement to modify a data item when you do not know the type of data (binary, packed, and so on) or its length. The MOVE statement takes care of all the details for you. Use the MOVE statement as follows:

1. To modify a data item where it is defined, type an m to the left of the statement defining it and press Enter. For example, to modify the value of TASKNUM, you would type in m instead of d.
2. If you type an \texttt{m} to the left of multiple lines, CA InterTest generates multiple \texttt{MOVE} statements.

3. To modify a data item where it is \textit{referenced}, type an \texttt{m} to the left of the statement referencing it, place the cursor under any character in the data item, and press Enter.

CA InterTest generates and displays a fill-in-the-blanks \texttt{MOVE} statement, as shown in the following screen:

4. Type one of the following items in the \texttt{MOVE} field:
   - A variable name, such as \texttt{NEXT-TASKNUM}
   - One of the following COBOL-like keywords: \texttt{ZEROS}, \texttt{SPACES}, \texttt{LOW-VALUES}, \texttt{HIGH-VALUES}, \texttt{QUOTES}
   - An alphanumeric literal enclosed in single quotes, such as 'YES'
   - A numeric literal, with or without a leading plus, +, or minus sign, -, such as -8.

5. Press Enter to execute the \texttt{MOVE} statement. CA InterTest displays a main storage display showing the new value of the data item. If you execute several \texttt{MOVE} statements, CA InterTest shows the main storage displays one at a time. Press Clear to see the next one.

6. From the main storage display, press Clear or PF3 to return to the source listing.

**Keeping Data Items in the Keep Window**

The Keep window lets you display the values of an unlimited number of data items directly on the source listing. If more than six variables are selected for the Keep window, PF19 and PF20 are available to scroll forward and backward throughout the Keep window. Keeping data items in the window lets you observe changes in their values as the program executes. You also can change values by overtyping the displayed bytes.
The Keep window consists of two parts. Dynamic keep elements are those that consist of the variables based on the current line of code. They are automatically displayed using the AutoKeep Display profile feature. They are shown in a highlighted fashion. Static keep elements are those that remain in the Keep window for the duration of the execution of the program until you remove them. The static keep elements are displayed before the dynamic elements. Use scrolling to view all the elements. You can change any dynamic element to a static element simply by following the rules mentioned next.

When the Keep window is active, the options and PF key functions are not displayed. Press PF4 if you need to refer to them. When all the data items in the Keep window are removed, the options and PF key functions are redisplayed. A command line is available to help you perform CA InterTest functions when the Keep window is active.

Add a Data Item to the Keep Window (static keep element)

You can add data items to the Keep window whenever your source listing is displayed. Identify the data items whose values you want to observe either before you execute the program, or when you are at a breakpoint.

Follow these rules to add a data item to the Keep window:

- To add a data item where it is defined, type a \texttt{k} to the left of the statement defining it, and press Enter.
- To add a data item where it is referenced, type a \texttt{k} to the left of the statement referencing it, place the cursor under any character in the item, and press Enter.

When you add a data item to the Keep window, CA InterTest responds by displaying the data item, as shown next.
Display the Data Item Structure

To display the entire COBOL structure of a data item, type a d to the left of the data item in the window and press Enter, as shown next.

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==> COBDEMO Option # Stmt # Margin= 01
______________________________
_____ EIBTRNID | DEMC
_____ EIBTRMID | U087
d____ TASKNUM | ?00000.
______________________________
000477 CONTINUE-TASK.
000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
A ==> ADD +1 TO TASKNUM.
```

CA InterTest responds by displaying the data item and all the items below it in the same COBOL structure.

Modify the Value of a Data Item

Modify the value of a data item in the Keep window by doing the following actions:

- Overtyping the displayed bytes, just as you would overtype the bytes in a main storage display
- Typing in m to the left of the data item to use the CA InterTest MOVE statement. Both methods of modifying a data item are discussed in the section, Inspecting a Modifying Main Storage.

**Note:** If you try to change password-protected storage areas not owned by your program, CA InterTest prompts you to enter the password.
Remove a Data Item from the Keep Window

To remove a data item from the Keep window, type an \texttt{x} to the left of the data item, as shown in the following screen:

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ==> Program= COBDEMO  Option #       Stmt #                             Margin= 01
Search=
-----------------------------------------------------------------------------------
| x| EIBTRNID                          | DEMC                      |
| x| EIBTRMID                          | U087                      |
|   | TASKNUM                          | ?00000,                   |
-----------------------------------------------------------------------------------
| 000477 CONTINUE-TASK.     |                               |
| 000478*** TASKNUM *NOTE* FIELD MUST BE INITIALIZED |                               |
A    ==>     ADD +1 TO TASKNUM.
```

In this example two data items (EIBTRNID and EIBTRMID) will be removed from the Keep window.

When all data items have been removed, the Keep window no longer displays.

Use Variable-Change Breakpoints to Detect Changing Values

You can request that CA InterTest for CICS halt execution whenever the value of a specified data item changes. This is called a variable-change breakpoint.

Set a Variable-Change Breakpoint

A quick way to set a variable-change breakpoint is from the point in the listing where it is defined. In the following example, you will set a variable-change breakpoint on TSQ-TERMID.
To set a variable-change breakpoint

1. Position the listing at the definition for TSQ-TERMID by typing `L TSQ-TERMID` in the Command line field, and press Enter.

2. The listing is then positioned at the definition for the TSQ-TERMID. Type `v` to the left of the statement defining the TSQ-TERMID field, position the cursor on the TSQ-TERMID field, and press Enter.
3. Type the DEMC transaction. The following breakpoint displays:

```
CA InterTest for CICS V9.1 - PROSYM FILE   COND BEFORE BREAKPOINT
COMMAND ==> COBDEMO Option # Stmt # Margin= 01
-- More:  + --------------- PKS 19 Backward 20 Forward -
    DMAP04AI            | ....................................
    DMAPBEGI            | .............................
    DMAPASRI            | .............................
    DMAPSUMI            | .............................
    DMAPENDI            | .............................
    DMAP00I
---------
+
-----------------------------------------------
| 000353   MOVE EXITMID TO TSQ-TERMID.
| 000354*** THIS CODE INITIALIZES THE MAPS BECAUSE COBOL2 DOESN'T ***
| V ==> MOVE LOW-VALUES TO DMAP04AI
|   ==> CAI6360 Conditional breakpoint requested at offset .ANY
|   ==> IF='TSQ-TERMID'.NE.'TSQ-TERMID'
|   ==> 000356   DMAPBEGI
| 000357   DMAPASRI
| 000358   DMAPSUMI
| 000359   DMAPENDI
| 000360   DMAP00I
```

**Note:** the listing is positioned on the statement following that which actually changed TSQ-TERMID, because the field is examined at the beginning of each new COBOL statement.

The listing is positioned on the statement following that which actually changed TSQ-TERMID, because the field is examined at the beginning of each new COBOL statement.

A variable-change-breakpoint may be set any time during execution, or from a listing of your program before execution begins. You may find it useful to also set a static keep on the field which you are interested in watching for a change. In this way you will be able see how it changed automatically each time a variable change breakpoint occurs.

Each variable change breakpoint is identified by the abbreviation VBP on the Status Report and Monitoring Status display.

Variable-change breakpoints are actually a special type of conditional breakpoint. CA InterTest for CICS checks for the condition at any statement, as shown in the following example:

```
IF 'data-item'.NE.data-item'
```

which means:

```
IF data-item (current) NOT EQUAL TO data-item (value when breakpoint set)
```

To explore additional ways to set and use conditional breakpoints, see the later chapters in this primer.
Remove a Variable-Change Breakpoint

From the variable-change breakpoint display, overtype the V with an x and press Enter to remove the breakpoint. You can also remove the breakpoint from the Monitoring Status display. Type an R next to the VBP option to be removed, and press Enter. The following screen shows removing the variable-change breakpoint set on the variable TASKNUM in the program COBDEMO.

---

Inspect and Modify Auxiliary Storage

You can use CA InterTest to inspect and modify the following items:

- VSAM and BDAM files
- DL/I, DB2, and SQL/DS databases
- Temporary storage
- Transient data

You can inspect and modify auxiliary storage at any time, even if a program is executing. This capability lets you maintain files and databases without writing one-time programs. You can perform any VSAM, BDAM, or DL/I function and most SQL functions. The ability to inspect and modify auxiliary storage when a program is stopped at a breakpoint lets you change test data in the middle of a test session.

Any changes you make are permanent; that is, changes to the file remain after the test session ends.
Inspect Auxiliary Storage

To access the auxiliary storage facility, select 4 Auxiliary storage from the CA InterTest Primary Option Menu, then specify the file you want to examine on the Auxiliary Storage Menu.

```
----------- CA InterTest for CICS V9.1 AUXILIARY STORAGE MENU -----------
OPTION ===> 1

Select an auxiliary storage type, specifying optional criteria below.

  1  Files       - Display/select files for access
  2  DB2 database - Invoke DB2 SQL interface facility
  3  DL/I database - Access DL/I database
  4  TD queues   - Display/select transient data queues for access
  5  TS queues   - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):
(Valid mask characters are * and/or +)

file____ ________ ________ _
  .
  .
  .
```

Alternatively, if you are at a clear CICS screen, you can invoke the FILE transaction. CA InterTest displays the following:

```
DATATYPE= FC FILEID=       MODE=      LOG=ON  TODEST=       PASSWORD=
FUNC=      SUBFUNC=      RETMETH=      ARGTYP=     SRCHTYP=
MESSAGE=
RETNRCID=
RCID=      CHGLEN=      DATA=        SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDEEFF *0123456789ABCDDEF+
LOC 0000       ........ ........ ........ ........ ........

CA InterTest for CICS V9.1

Copyright © 2014 CA. All rights reserved
```

**Note:** If the FILE transaction does not display the previous screen, it is probably because the person who installed CA InterTest changed the name of the FILE transaction. Contact that person to find out the correct name.
After you type the file or database and, optionally, the record or segment, CA InterTest displays the data. A sample VSAM record in dump format follows.

```
DATATYPE= FC FILEID= INVNTRY   MODE=     LOG=OFF  TODEST=        PASSWORD=
FUNC=     SUBFUNC=      RETMETH=        ARGTYP=    SRCHTYP=
MESSAGE= RECORD OBTAINED FOR VIEWING
RCIID=C 0000000005'  DATA=
FORMAT= D 0012233 44556677 8999AABB CCDEEFF *0123456789ABCD* 0000000005
LOC 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000    F0F0F0F0 F0F0F0F0 F0F54040 0087338F 0000000005       DSORG=VSKS
0020    40404040 40404040 C1C2C340 C3D6D9D7           ABC CORP  LRECL=0050
0030    F0F9F1F2 F8F74040 0000598C 40404040   091287
0040    D1D6C3D5 404040E2 D4C9E3C8 40404040   JOHN  SMITH      KEYPOS=0000
0050                                                           KEYLEN=0A
0060                                                           STRNO=01
0070
0080 0090 00A0                                          READ  ADD  UPDATE
-------------------------------------------------------------------
1 Help       2 Format C   3 End        4 BEGB       5 PREV      6 DataType DL
7 Page bwd   8 Page fwd   9 Caps Off 10 Top 11 Bottom 12
```

**Note:** The data appears in both hexadecimal and character format.

To return to the source listing, press Clear or PF3.

**Modify Auxiliary Storage**

To modify the record or segment, overtype the hexadecimal or character data and press Enter. For example, you could overtype JOHN SMITH with DALE COOPER on the sample display in the previous figure.

To rewrite the modified record, type **PUT** in the **FUNC=** field and press Enter.
Display Auxiliary Storage in Structured Format

CA InterTest can display file, transient data, and temporary storage records or DL/I segments field-by-field in structured format using 01 level data names.

To display a record or segment in structured format, you must identify the program containing the structure. Symbolic information for this program must be saved in the CA InterTest Symbolic File. Follow these steps to display a record or segment in structured format:

1. If the FORMAT field does not contain S, press PF2 until FORMAT= S is displayed. The USE=field now appears after the FORMAT field.
2. Identify the program and structure in the USE= field. Specify the command as follows:
   USE=symbolic-name.structure-name
   *symbolic-name* is the name of the program as defined in the CA InterTest Symbolic File
   *structure-name* is the COBOL 01 level name
3. Press Enter. CA InterTest displays the record or segment in structured format. A sample VSAM record in structured format is shown next.

Structured format lists the contents of each data name in both hexadecimal and character format.

4. To modify the record or segment, overtype either the hexadecimal or character data and press Enter. To rewrite the modified record, type PUT in the FUNC= field and press Enter.
When a program is stopped at a breakpoint, you can substitute an asterisk (*) for the symbolic-name. For example, if program PAYROLL is stopped at a breakpoint and you type:

USE=* .payrec

CA InterTest uses the program named PAYROLL and the PAYREC structure.

To begin the structure with a specific data name, specify the following command in the DATA field:

FIND=data-name

where data-name is the data name to be displayed.

Resume Program Execution

When you are finished performing breakpoint activities such as displaying and modifying main or auxiliary storage, you can resume program execution. You can do the following actions:

- Resume execution from a specific location
- Single-step execution one verb at a time

Resume Execution from a Specific Location

To resume program execution from the breakpoint location where your program is stopped, the source listing screen must be displayed. Press PF3 from any CA InterTest screen until the breakpoint screen redisplays.

Press PF5 to continue from the current breakpoint. Execution continues until the program is halted at another breakpoint or the task completes.

If you are stopped at an automatic breakpoint, you must first correct the error that triggered the breakpoint before continuing execution from that point. Or, you can go around the error and continue execution from another location.

To resume program execution from another location, perform the following steps:

1. Display the statement where you want to resume execution. For more information, see Displaying Your Source Listing.
2. Type go read-dataset in the command line or g to the left of that statement.
3. Press Enter.

Program execution resumes from that statement.
The following example shows how to resume execution from a location prior to an automatic breakpoint. This might be useful if you corrected a record or dynamically modified an incorrect file name and now want the program to re-read the record or file.

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ===>
Program= COBDEMO Option # Stmt # Margin= 01
Search=

| RECORD-KEY | 00000000000000000000000000000000000000 |
|---------------|
| 000695 | GO TO SEND-REWRITE-RETURN. |
| 000696 | IF TASK-SWITCH3 EQUAL 'A' |
| 000697 | GO TO READ-DATASET |
| 000698 | ELSE |
| 000699 | GO TO SEND-MAP00. |
g 000700 | READ-DATASET. |
| 000701 | EXEC CICS HANDLE CONDITION |
| 000702 | DSIDERR |
| 000703 | NOTOPEN(NOT-OPEN) |
| 000704 | END-EXEC. |
| 000705 | Call 'DFHEI1' using by content x'0204800007130c0000000000000000' |
| 000706 | '0000000000000000f0f0f3f8f7404040' end-call |
```

When you press Enter, execution resumes from READ-DATASET, the location you identified with g.

**Single-Stepping**

Single-stepping lets your program execute one COBOL verb and then stop again. You can adjust the step amount so the program executes a specified number of verbs between stops.

To single-step, type `next` in the command line and press Enter or press PF10 from the source listing. Your program executes the next verb and then stops.

To single-step from a different program location:

1. Display the statement where you want to resume execution. For more information, see [Display Your Source Listing](#) (see page 36).
2. Type a g to the left of that statement.
3. Type `next` in the command line and press Enter or press PF10.
To adjust the step amount, perform the following steps:

1. Press PF4 to access the Source Listing Profile screen.
2. Change the number in the Stepping amount field.
3. Press Enter. CA InterTest changes the step amount and redisplays the source listing. If Titles are set to on (an option on the Source Listing Profile screen), you see the new step amount displayed next to PF10 at the top of the Source Listing Breakpoint screen.

**Abend a Task**

When your program is stopped at a breakpoint, you can abend your task rather than resume execution. Type `abend` in the Command line of any Breakpoint display. CA InterTest displays the screen shown next.

```
-------------------------- CA InterTest V9.1 ABEND BREAKPOINTED TASK --------------------------
COMMAND ===> 

Type an abend code, then press ENTER.

Abend Code ____ Abend code options are:

  blanks    Normal abend, no dump
  XXXX      Abend exits cancelled, no dump
  your code Your abend code, dump taken

■ To cancel the abend, press PF3. CA InterTest redisplays the Primary Option Menu.
■ To abend *without* a dump:
  – For a normal abend, press Enter.
  – To cancel all program exits, specify code XXXX and press Enter.
■ To abend *with* a dump, specify a four-character dump code and press Enter. The dump code you specify cannot be XXXX and cannot begin with the letter A.

For more information on the status of your program after abending, see Ending a Test Session.

**Get Help**

Help is always available when you are using CA InterTest. You can use Help to initiate CA InterTest. Type `Help` on a clear screen under CICS.
Get Help in Using CA InterTest for CICS

When you are using CA InterTest, you can press PF1 at any time for information on CA InterTest functions. For example, the following screen lists the Help topics for the Source Listing facility.

```
CA INTERTEST - INTERACTIVE HELP FACILITY - (V9.1)
TUTORIAL: SOURCE LISTING FACILITY - FOR COBOL PROGRAMS

05  Set Monitoring                        45  Set/Remove other monitoring Options
10  Locating areas in the listing         50  Displaying data items
20  Search for a Character string         55  Modify the value of data items
25  Specify Indirect Commands            60  Resume TASK execution
30  PF key definitions                    65  Abending your TASK
32  Primary Line Commands                 70  PROFILE, AUTOSTEP, AUTOKEEP Options
35  Set the Display Margins               75  Code Coverage Option
40  Set/Remove Breakpoints                80  View Program Backtrace

(end)
```

ENTER A SELECTION CODE FROM THE MENU, OR N FOR NEXT PAGE, P FOR PRECEDING PAGE, M FOR RETURN TO PREVIOUS MENU, CLEAR TO EXIT, OR MESSAGE NUMBER. ==> 05

Help for Line Commands

Notice that selection code 32 lists Line Commands. These are the simple commands that you can enter on the Command line of a Source Listing Display, Breakpoint Display, or CA SymDump Breakpoint display, and include the commands abend, monitor, and status.
Get Help to Correct an Error

Whenever your program is stopped at an automatic breakpoint, CA InterTest displays a short message identifying the reason for the breakpoint. You can request help by pressing PF1 for information on what caused the error that triggered the breakpoint and how to fix it. The following screen explains how to correct an AEIL abend:

```
CA INTERTEST - INTERACTIVE HELP FACILITY -(V9.1)
TUTORIAL: ERROR MESSAGE AEIL

The dataset name referred to in the DATASET option cannot be found in the FCT.
Your program did not have a Handle Condition for this error. Or a second Handle
Condition without a routine for this error superseded the one that had a
routine for this error.

WHAT YOU CAN DO: If the named file was entered incorrectly and the one you
wanted exists, you may use the Replace File Option to dynamically replace the
file name and then use the resume task facilities to execute the CICS request
again. To perform the above functions from the Source Listing Breakpoint screen
you would:

1. Key =20s in the Option # field and press ENTER.
2. Tab to Replace file name: Key in the incorrect file name.
3. In the next field, key in the correct file name and press ENTER.
4. Press CLEAR to return to your Source Listing Breakpoint screen.
5. Key in G to resume execution at the beginning of the EXEC CICS command
   and press ENTER.
```

Note: When you leave and then return to an automatic breakpoint screen, the error
message may no longer appear. For help on the cause of the error, press Clear to
redisplay the error message before pressing PF1.

End a Test Session

After your program completes execution or you abend your task, you return to CICS. The
following table shows the status after testing of a program area:

<table>
<thead>
<tr>
<th>Program Area</th>
<th>Status after Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>program source code</td>
<td>not modified</td>
</tr>
<tr>
<td>file/database updates by the program</td>
<td>permanent</td>
</tr>
<tr>
<td>file/database updates you made</td>
<td>permanent</td>
</tr>
</tbody>
</table>

CA InterTest monitoring, breakpoints, and other options set for the program remain in
effect until they are specifically removed.
Correct the Source Code

At some point you might want to correct the source code and continue testing from a clean version of the program.

After you have modified the source code and recompiled the program, you can load the new copy of the program by using the New Program Copy option on the Program Monitoring menu (ITST 2.1). The next chapter explains how to set options on the Program Monitoring menu.

The New Program Copy option resets symbolic breakpoints and other monitoring options for the recompiled program. It also resets the PPT entry to the program's new library address.
Chapter 4: Advanced Monitoring Features

CA InterTest for CICS has many advanced features that help you detect and correct errors. Whatever your specific testing needs, CA InterTest has the necessary tools.

This chapter describes just a few of the more commonly used features; for information on all of CA InterTest's features, see the User Guide and the Help facility.

This section contains the following topics:

- Set Options from the Monitoring Menus (see page 63)
- Set and Remove Conditional Breakpoints (see page 65)
- Remove Conditional Breakpoints from the Source Listing (see page 67)
- Request Breakpoints (see page 68)
- Backtrace Facility (see page 71)
- Set the Code Counting/Coverage Option (see page 75)
- Statement Trace Facility (see page 76)
- Indirect Commands (see page 79)
- Replacement, Protection, and Special Options (see page 84)
- Composite Module Testing (see page 86)
- Dump Analysis with CA SymDump (see page 87)

Set Options from the Monitoring Menus

The CA InterTest monitoring menus are the easiest way to set many monitoring options. You can access these menus from the Primary Option Menu by selecting 2 Monitoring, as shown in the following screen:

```
-------------- CA InterTest for CICS V9.1 PRIMARY OPTION MENU --------------
OPTION ======> 2
  1 Source - Display/select program source files/listings
  2 Monitoring - Display/modify CA InterTest monitoring/activity
  3 Main storage - Display/modify CICS storage areas
  4 Auxiliary storage - Display/access databases/files/queues
  5 Dump analysis - Invoke CA SymDump CICS dump/trace capture facility
  6 Product help - Invoke CA InterTest product help facility
  7 Status/Maintenance - Product status and maintenance functions
  8 What's new? - Display information about CA InterTest V9.1
  X Exit - Terminate menu processing
```
CA InterTest displays the Monitoring Menu (shown next), from which you can select a wide variety of different monitoring options. Use online help or the User Guide for a detailed explanation of each of these options.

---

**CA InterTest for CICS V9.1 MONITORING MENU**

**OPTION :=>**

1. Programs - Display/modify program monitoring options
2. Transactions - Display/modify transaction monitoring options
3. Terminals - Display/modify terminal monitoring options
4. Status - Display/remove current monitoring options
5. Active tasks - Display/purge active monitored tasks
6. System-wide - Display/modify global system-wide options

---

Option 1 Programs access the Program Monitoring Menu, from which you can set and remove any option for a program, and request a status display.

The Program Monitoring menu, shown in the following screen, is probably the most frequently used menu during a testing session:

---

**CA InterTest V9.1 PROGRAM MONITORING**

**COMMAND :=>**

Type information and S to set or R to remove option(s) below.

<table>
<thead>
<tr>
<th>Program . . cobdemo</th>
<th>Program name (or .ALL, .OPTIONS or generic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID . . _______</td>
<td>User (or .ANY) for whom the program is monitored</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>Status</td>
<td>Display and/or remove monitoring options (S only)</td>
</tr>
<tr>
<td>Monitor</td>
<td>Monitoring (R removes monitoring and all options previously set)</td>
</tr>
<tr>
<td>UBP</td>
<td>Unconditional breakpoints (specific program only)</td>
</tr>
<tr>
<td>CBP</td>
<td>Conditional breakpoints (specific program only)</td>
</tr>
<tr>
<td>RBP</td>
<td>Breakpoints for CICS, DB2, DL/I or external CALL requests</td>
</tr>
<tr>
<td>Stmt Trace</td>
<td>Statement tracing and data monitoring (COBOL only)</td>
</tr>
<tr>
<td>New copy</td>
<td>Fetch new copy of program and reset monitoring options (S only)</td>
</tr>
<tr>
<td>Commands</td>
<td>Indirect commands defined for a specific COBOL or PL/1 program</td>
</tr>
<tr>
<td>Replace</td>
<td>CICS resource name replacement options</td>
</tr>
<tr>
<td>Protect</td>
<td>Storage protection monitoring options</td>
</tr>
<tr>
<td>Special</td>
<td>Other options (storage allocation, file updating, etc.)</td>
</tr>
<tr>
<td>Composite</td>
<td>Monitor multi-CSECT program’s separately compiled components</td>
</tr>
</tbody>
</table>

**PF1 Help** 2 3 End 4 Return 5 6
**PF7 Backward** 8 Forward 9 10 11 12

Quick access to this menu is available before or during a session, as described in the following table. Type the fastpath entry in the indicated field and press Enter.

<table>
<thead>
<tr>
<th>From</th>
<th>Fastpath Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS, clear screen</td>
<td>ITST 2.1</td>
</tr>
</tbody>
</table>
Set and Remove Conditional Breakpoints

Conditional breakpoints are breakpoints that take effect when a specific condition is met, such as when a counter exceeds a value. These breakpoints can help you isolate complex problems.

The previous chapter showed how easy it is to set an unconditional breakpoint directly on your source listing. All you do is type a **u** to the left of the statement where you want to set the breakpoint.

To set a conditional breakpoint, type a **c** to the left of the statement where you want to set the breakpoint, as shown next, and press Enter.
CA InterTest displays the menu on which you define the condition. For the following example, the breakpoint halts program execution only if TASK-SWITCH is equal to 3.

```
CA-InterTest MONITORING COMMAND BUILDER - CONDITIONAL BREAKPOINT

Enter LEFT SIDE
  Data Name task-switch

Enter OPERATOR (EQ, NE, GT, LT, GE, LE): eq

Enter RIGHT SIDE
  Data Name
  OR
  Literal 3

  ENTER S to Drop monitoring on a true condition

For location: 000498
  MOVE SPACE TO TASK-SWITCH.

Press PF9 to go to complex conditional screen if necessary
PF1 Help  2  3 End  4 Return  5  6
PF7  8  9 Complex  10

Conditional breakpoints are indicated by a C next to the line of code in your source listing.
```
Remove Conditional Breakpoints from the Source Listing

You can easily remove a conditional breakpoint from where it is displayed in your source listing or from the monitoring status report. To remove a conditional breakpoint from your source listing, overtype the C with an x and press Enter, as shown in the following screen:

```
CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> Option #       Stmt #       Margin= 01
Program= COBDEMO   Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map 5 Clst/Pmap More: +
 6 Data xref 7 Proc xref 8 Err msgs 9 Srch fwd 10 Srch bwd
PFKS 1 Help 2 3 End 4 Profile 5 Monitor 6 Menu
 7 Backward 8 Forward 9 Next Wnd 10 11 12 Status
-------------------------------------------------------------------------------
_ 000435       Go to GEN-ERR depending on dfheigid.
 000436
 000437 NO-OPTS.
 x 000438       MOVE SPACE TO TASK-SWITCH.
 000439
 000440
```
Remove Conditional Breakpoints from the Status Report

You can always remove any breakpoint from the Monitoring Status report. To remove this conditional breakpoint from your program, perform the following steps:

1. Type **STATUS** on the Command line to display the Monitoring Status report for the current program.
2. Tab to the entry for the Conditional Breakpoint (CBP option) you want to remove, type an **r** to the left of its entry, and press Enter.

   CA InterTest places an asterisk to the left of the breakpoint to indicate that the removal was processed.

3. Press PF3 to return to your Source.

Request Breakpoints

Request breakpoints are breakpoints that take effect prior to CICS commands and macros and other program calls, such as calls to DL/I or DB2. With one specification, you can set breakpoints at all CICS commands or at specific commands, such as all REWRITE commands.

Use request breakpoints to stop a program before the following commands:

- Every CICS command or macro
- Specific types of CICS commands, such as all File Control or Program Control commands
- Specific commands, such as all READ or WRITE commands
Set Request Breakpoints

To set request breakpoints, perform the following steps:

1. Select option 2 Monitoring on the Primary Option Menu. CA InterTest displays the Monitoring Menu.

2. Select option 1 Programs. CA InterTest displays the Program Monitoring screen.

3. In the Program field, type the name of the program for which you want to set request breakpoints.

4. Type an s to the left of the RBP option (request breakpoints). Press Enter.
CA InterTest displays the Request Breakpoint Selection menu on which you specify the type of request breakpoint, such as: all CICS commands, File Control and Task Control.

![Request Breakpoint Selection Menu]

Subsequent menus prompt you for the specific command, such as all File Control commands, READ, WRITE, REWRITE, and so on.

**Note:** Type an x in front of each type of command for which you want to set request breakpoints.

After setting your request breakpoints, PF4 returns you to the Primary Option Menu. If you came from a Source Listing, PF3 returns you to your source.

### Remove Request Breakpoints

To remove request breakpoints, perform the following steps:

1. From the Source Listing Display screen or breakpoint display, type **STATUS** on the Command line and press Enter.

   CA InterTest displays the Monitoring Status display showing all options for the current program.

2. Type an r to the left of the request breakpoint (RBP) you want to remove and press Enter. CA InterTest places an asterisk to the left of the breakpoint to indicate that the command has been processed.

3. Press PF3 to return to the Source Listing screen or Breakpoint display.
Backtrace Facility

When your program stops at a breakpoint, you might find it useful to examine the Backtrace Summary and the Source Listing Backtrace. The backtrace facility shows the logic flow of the program and explains how processing reached this point.

The backtrace facility lets you do the following actions:

- View the backtrace summary screen for a (high-level) summary of the program’s execution
- Use special PF keys to step through the program's execution path
- Reposition the source listing to view the program’s execution path from a different backtrace position

In addition to the Backtrace PF keys, all other source listing options, such as the Keep window, setting and removing breakpoints, and the ability to scroll backward and forward from the current source listing position are available from the Source Listing Backtrace screen.
Access the Backtrace Summary

To view the Backtrace Summary, press PF11 from the Source Listing Breakpoint screen, or, type BTRACE in the Command line and press Enter. CA InterTest positions the Backtrace Summary Screen at the last executed statement, as shown in the following screen:

CA InterTest for CICS V9.1 - BACKTRACE SUMMARY

Program= COBDEMO From 0005 To 0017 Of 0017

Specify S then ENTER to display Source Listing BACKTRACE

PFKS 1 Help  2 Backtrace  3 End  4 5 1st Stmt  6 Last Stmt
7 Backward  8 Forward  9 Next Wnd  10  11 Prev Bloc  12 Next Bloc

<table>
<thead>
<tr>
<th>S Bkmk</th>
<th>Stmt Block</th>
<th>Source Listing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#375.0...#378.0</td>
<td>Call 'DFHEII' using by content x'0204c000072c0</td>
</tr>
<tr>
<td></td>
<td>#378.0...#378.0</td>
<td>Go to WRITE-TSQ GEN-ERR depending on dfheigd</td>
</tr>
<tr>
<td></td>
<td>#407.0...#407.0</td>
<td>Call 'DFHEII' using by content x'02048000071b0</td>
</tr>
<tr>
<td></td>
<td>#407.0...#410.0</td>
<td>Call 'DFHEII' using by content x'02048000071b0</td>
</tr>
<tr>
<td></td>
<td>#416.0...#416.0</td>
<td>Call 'DFHEII' using by content x'0e06c00007000</td>
</tr>
<tr>
<td></td>
<td>#416.0...#426.0</td>
<td>Call 'DFHEII' using by content x'0e06c00007000</td>
</tr>
<tr>
<td></td>
<td>#426.0...#432.0</td>
<td>Call 'DFHEII' using by content x'0e06a00007000</td>
</tr>
<tr>
<td></td>
<td>#432.0...#435.0</td>
<td>Call 'DFHEII' using by content x'02048000071b0</td>
</tr>
<tr>
<td></td>
<td>#438.0...#447.0</td>
<td>MOVE SPACE TO TASK-SWITCH.</td>
</tr>
<tr>
<td></td>
<td>#447.0...#460.0</td>
<td>Call 'DFHEII' using by content x'0a02e00007000</td>
</tr>
<tr>
<td></td>
<td>#460.0...#468.0</td>
<td>Call 'DFHEII' using by content x'1804f00007000</td>
</tr>
<tr>
<td></td>
<td>#468.0...#470.1</td>
<td>Call 'DFHEII' using by content x'0402000007000</td>
</tr>
<tr>
<td>*</td>
<td>#479.0...BKPT</td>
<td>ADD +1 TO TASKNUM.</td>
</tr>
</tbody>
</table>

Read the Backtrace Summary

The Backtrace Summary screen summarizes the program’s execution. This screen displays the statement number and source of the first statement in each contiguous group of executed program statements.

- The order of the statements reflects the program’s execution path
- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

From the Backtrace Summary screen, you can do the following actions:

- Reposition the display to view statement-by-statement details of the program's execution path by:
  - Pressing PF2
  - Marking a statement block with s, then pressing Enter
Assign a *bookmark*, consisting of one to four characters, to one or more backtrace positions for faster and easier navigation through the Backtrace Summary.

Trace program execution using the following PF keys:

- **PF5 1st Stmt**: Repositions the Backtrace Summary screen to the 1st or oldest entry in the Backtrace.
- **PF6 Last Stmt**: Repositions the Backtrace Summary screen to the last or newest entry in the Backtrace.
- **PF7 Backward**: Scrolls the Backtrace Summary screen backward (up) a full screen.
- **PF8 Forward**: Scrolls the Backtrace Summary screen forward (down) a full screen.
- **PF11 Prev Bloc**: Displays the previous statement block.
- **PF12 Next Bloc**: Displays the next statement block.

**Access the Source Listing Backtrace**

To display the Source Listing Backtrace screen, press PF2 or type s next to a statement block in the Select column of the Backtrace Summary screen, as shown in the previous screen. The following screen is the Source Listing Backtrace screen.
Read the Source Listing Backtrace

To clearly detail the program's execution path, statement by statement, source statements that were executed from the current backtrace position are highlighted. When you specify a non-backtrace source listing option or use PF7 or PF8 to reposition the source listing, the highlighting of executed statements is temporarily suspended until a backtrace-related PF key (next block, previous block, and so on) is entered.

When in source listing backtrace mode, the current backtrace position always appears on the line just above the first source statement. This line displays as follows:

======>BACKTRACE AT #nnnnn (nnnnn -> nnnnn executed nnnnn times)<=====

The information in this line indicates the following:

- #nnnnn: The program statement or offset the backtrace is currently positioned at
- nnnnn -> nnnnn: The first and last number of the statement block
- nnnnn times: How many times a statement block was executed

On the Source Listing Backtrace screen, each relevant backtrace position is indicated in the listing by an arrow or a line, as shown in the following table:

<table>
<thead>
<tr>
<th>Backtrace Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=====&gt;</td>
<td>Indicates the first statement in a statement block</td>
</tr>
<tr>
<td>&lt;=====</td>
<td>Indicates the last statement in a statement block</td>
</tr>
<tr>
<td>======</td>
<td>Indicates all other statements within a statement block</td>
</tr>
</tbody>
</table>

The Source Listing Backtrace screen displays the source of each statement:

- The order of the statements reflects the program's execution path.
- The top statement is the first statement executed; the bottom statement is the most recently executed statement.

Navigate through Program Execution

You can trace your program's execution path by using the following commands:

- Using PFS, PF9 and PF11 to trace execution backward
- Using PF6, PF10, PF12 to trace execution forward
- Setting the source listing profile Stepping Amount, and then using PF9 and PF10 to automatically trace the execution backward or forward
End a Source Listing Backtrace Session

To turn off the source listing backtrace display mode, press Clear or PF3. CA InterTest redisplay the Source Listing Breakpoint screen, positioned at the last breakpoint.

To toggle between the Source Listing Backtrace screen and the Backtrace Summary screen, press PF2.

Set the Code Counting/Coverage Option

CA InterTest has many options for specialized testing needs. The code coverage option lets you see the number of times an assembler statement was executed.

The initial setting for code counting is OFF, which is the default. However, you can use the Source Listing Profile screen to turn it ON or OFF again at any time after you have selected monitoring for a program.

How the counts are displayed or not displayed is handled by the primary line command COUNTS and its associated parameters.

This feature causes overhead during program monitoring and should be turned off as soon as it is no longer needed.

To change the setting for code coverage, perform the following steps:

1. Type profile on the command line and press Enter, or press PF4 to access the Profile screen from any Source Listing screen. The Code Counting= field shows the current setting.

2. Overtype the current value with one of the following values:

   YES  Enables code counting and turns on the COUNTER display feature.
   NO   Removes the COUNTER display and stops the counting feature.

3. Press Enter to process the new settings and to return to the Source Listing Display screen.
The following screen details the code coverage feature:

<table>
<thead>
<tr>
<th>TASKNUM</th>
<th>COUNTER</th>
<th>Command</th>
<th>Stmt #</th>
<th>Margin</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td></td>
<td>PROCEDURE DIVISION</td>
<td></td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>using dfheiblk dfhcommarea</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>MOVE EIBTRNID TO TSQ-TRANID.</td>
<td>0000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>MOVE EIBTRMDID TO TSQ-TERMID.</td>
<td>0000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>THIS CODE Initializes THE MAPS BECAUSE COBOL2 DOESN'T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>MOVE LOW-VALUES TO DMAP04AI</td>
<td>0000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP00G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP01I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP02I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP03I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP04I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP05I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP06I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td>DMAP07I</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Statement Trace Facility**

The Statement Trace Facility lets you view the logic flow of your program at a statement level. When used in conjunction with the Datamon command, it allows you to view the values of data items at the time that the statement was executed.

**Note:** This feature is only valid for COBOL programs. Use the Backtrace facility for tracing information for other languages.

**Enable Statement Tracing and Data Monitoring**

Use the TRACE command to enable the statement tracing facility. This creates a table whose number of entries is defined by the STMTTRACE option in IN250OPTS. This is a wraparound table, meaning that when all of the entries are used, the oldest entry is used for the current statement.

Use the DATAMON command, or DM, to enable data monitoring for the given program. If the Statement Trace facility was not in effect when data monitoring is set, a statement trace table is created.
Navigate the Statement Trace Table

When at a breakpoint, and statement tracing is in effect, you can use the PREV and ADVANCE commands to determine the execution path of the program prior to this point. Use the PREV command to cause the display to back up one statement. Consider the following breakpoint:

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ===> PROTSYM FILE ABEND DETECTED BREAKPOINT
Program= COBDEMO Option # Stmt # Margin= 01
Search=

---------+-----------------------------------------------------------
_ 000477 CONTINUE-TASK.
_ 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
  A  ==> ADD +1 TO TASKNUM.
  ==> ASRA ABEND (0C7) detected and prevented. Caused by invalid decimal
  ==> arithmetic data format.
  ==>  Press PF1 for a detailed description.
  ==>  000480 IF TASKNUM = 1
  _ 000481   MOVE 'DMAPASR' TO MAPNAME.
  _ 000482   IF TASKNUM = 2
  _ 000483   MOVE 'DMAPSUM' TO MAPNAME.
  _ 000484   IF TASKNUM GREATER 2
  _ 000485   GO TO SEND-EN-D-MSG.
_ 000486   GO TO REWRITE-TSQ.
_ 000487 REWRITE-TSQ.
```
Using the PREV command on the previous screen results in the next screen:

```
CA InterTest for CICS V9.1 - PROSYM FILE  ABEND DETECTED BREAKPOINT
Command ===> COBDEMO  Option #  Stmt #  Margin= 01
Search=
OPTS 1 Proc div 2 Work-stor 3 Link sect 4 D-map  5 Clst/Pmap More: +
6 Data xref 7 Proc xref 8 Err msgs 9 Srch f wd 10 Srch bwd
PFKS 1 Help  2 3 Det Bkpt 4 Profile 5 RESUME 6 Menu
7 Backward 8 Forward 9 Next Wnd 10 001 Verb 11 Backtrace 12 Status
--------------------------------------------------------------------------
_ 000468 Call 'DFHEI11' using by content x'040200000700000014000040000000'
_ 000469 - '00f0f0f2f0f9404040' end-call.
_ 000470 IF EIBAID = DFHEI11 GO TO CONTINUE-TASK.
_ 000471 IF EIBAID = DFHECLEAR GO TO SEND-END-MSG.
_ 000472 IF EIBAID = DFHPF3  GO TO SEND-END-MSG.
_ 000473 IF EIBAID = DFHPF15 GO TO SEND-END-MSG.
_ 000474 IF EIBAID = DFHPF2  GO TO EXPANDED-DEMO.
_ 000475 IF EIBAID = DFHPF14 GO TO EXPANDED-DEMO.
_ 000476 GO TO SEND-FIRST-SCREEN.
_ 000477 CONTINUE-TASK.
_ 000478**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
    A ===> ADD +1 TO TASKNUM.
_ 000480 IF TASKNUM = 1
_ 000481    MOVE 'DMAPASR' TO MAPNAME.
_ 000482 IF TASKNUM = 2
_ 000483    MOVE 'DMAPSUM' TO MAPNAME.
```

Statement number 470 was the statement executed just before the abend at the statement number 479. Similarly the ADVANCE command advances the statement trace pointer.

**View Past Data Values**

When data monitoring is enabled, the user is able to view past data values when navigating the trace table. Using the PREV and ADVANCE commands, data item values reflect the values from when that statement was executed. These values are protected—you cannot update data items while viewing past data values.

**Usage Notes**

The DATAMON command can be very CPU intensive—it requires a significant amount of storage. Therefore, this command should be used with caution.
Indirect Commands

The CA InterTest Indirect Commands facility lets you specify a set of indirect commands to be executed in one of two ways:

- Automatically, when a breakpoint pointing to that set of commands is reached
- On demand, when you are at a breakpoint and use the Breakpoint Primary Option menu to resume execution with an indirect command (the fastpath is =4.5 from any breakpoint)

By using indirect commands, you can dynamically modify or correct programs without having to leave your test session to recompile your program.

With the indirect commands facility, you can do the following actions:

- Change the flow of control in your program
- Test conditions based on specified variables
- Change the value of specified variables
- Create and execute commands as a group, like adding a new subroutine
- Automatically resume execution of your program at the same or different location
- Define abbreviations for variables with long names

Code Indirect Commands

CA InterTest provides an indirect commands facility for coding the statements. You can access it from the source listing or from the ITST menus.

1. To access the indirect commands facility from the Source Listing Display screen, type icmds in the command line or 11 in the Option # field and press Enter, as shown in the following screen:

```
CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND === icmds
Program= COBDEMO Option # Stmt # Search= Margin= 01
--------------------------------------------------------------------------------
00001 ID DIVISION.
00002 PROGRAM-ID. COBDEMO.
00003 ENVIRONMENT DIVISION.
00004 DATA DIVISION.
00005 WORKING-STORAGE SECTION.
  00006 77 S999-FIELD1 PIC S9(3).
  00007 77 S999-FIELD2 PIC S9(3) VALUE +50.
  00008 77 999-FIELD1 PIC 9(3).
  00009 77 999-FIELD2 PIC 9(3) VALUE 50.
  .
  .
```
To access the Indirect Commands facility from the menus, access the Program Monitoring menu (fastpath is `ITST 2.1`), and select the Commands option. Perform the following steps:

1. Select option **2 Monitoring** on the Primary Option Menu. CA InterTest displays the Monitoring Menu.
2. Select option **1 Programs**. CA InterTest displays the Program Monitoring screen.
3. In the Program field, type the name of the program for which you want to create indirect commands.
4. Type an `s` to the left of the Commands option (Indirect commands), as shown in the following screen. Press Enter.

```
------------------ CA InterTest V9.1 PROGRAM MONITORING ------------------
COMMAND =>

Type information and S to set or R to remove option(s) below.

Program . . cobdemo_  Program name (or .ALL, .OPTIONS or generic)
User ID . . _______  User (or .ANY) for whom the program is monitored

Option            Description
Status             Display and/or remove monitoring options (S only)
Monitor            Monitoring (R removes monitoring and all options previously set)
UBP                 Unconditional breakpoints (specific program only)
CBP                 Conditional breakpoints (specific program only)
RBP                 Breakpoints for CICS, DB2, DL/I or external CALL requests
Stmt Trace         Statement tracing and data monitoring (COBOL only)
New copy           Fetch new copy of program and reset monitoring options (S only)
Commands           Indirect commands defined for a specific COBOL or PL/1 program
Replace            CICS resource name replacement options
Protect            Storage protection monitoring options
Special             Other options (storage allocation, file updating, etc.)
Composite          Monitor multi-CSECT program’s separately compiled components

PF1 Help  2  3  End  4  Return  5  6
PF7 Backward  8  Forward  9  10  11  12
```

Either access method takes you to the Indirect Commands screen shown in the following example.

This example shows the indirect commands that you use to fix the ASRA condition in our COBDEMO demo program, without recompiling. Setting TASKNUM to 1 alters the demo results.

Use the clear key to complete the termination of the demo when the "You have completed the sample CA InterTest test session" screen appears.
To code the indirect commands for your own programs, perform the following steps:

1. Specify the terminal ID (terminal ID, .ANY or .NO) where the indirect commands will take effect. The default terminal varies according to how you are monitoring the program.
   - If you are monitoring under a User ID of .ANY, the default terminal is the terminal you are using to specify the indirect commands.
   - If you are monitoring under your own User ID, the default terminal is .ANY.

2. Type the statement numbers and the text of the indirect commands as shown previously on the Indirect Commands screen. For more information, see Indirect Commands in the User Guide and online Help.

3. After coding the statements, press PF3 to exit the Indirect Commands facility. If you accessed the facility from the ITST menus, use =x to exit the menus.
How to Control the Flow of Execution

When you run your program, if a breakpoint points to indirect commands, the commands are automatically executed without stopping first. Execution is returned to the program according to the exit command that you specified in your indirect commands.

To return execution to your program following indirect command processing, use one of the following exit commands:

- **BREAK**—CA InterTest halts indirect command processing and issues a breakpoint display from where the indirect commands were invoked in the program.
- **GOTO**—CA InterTest resumes execution at the program statement number, offset, paragraph name, or indirect command statement number specified after the GOTO command.
- **EXIT**—CA InterTest resumes program execution at the breakpoint location from which the indirect commands were invoked and continues debugging the program until the next breakpoint.
- **RUN**—CA InterTest resumes program execution at the breakpoint location from which the indirect commands were invoked and ignores all subsequent breakpoints.

Attach the Indirect Command to Breakpoints

To have the indirect commands automatically executed by your program, set an unconditional, conditional, or request breakpoint that points to the first indirect command statement number. When you set a breakpoint with this option, each time the breakpoint takes effect the indirect commands are executed.

From the Source Listing Display screen, set the unconditional or conditional breakpoint as follows:

1. Display the line in your source code where you want to set the breakpoint to execute the indirect commands.
2. Type `bpo` in the command line or `12` in the Option # field (Breakpoint Options), tab down to the line where you want to set the breakpoint, type `u` for unconditional or `c` for conditional, and press Enter.

   The Breakpoint Locations menu displays, which lets you set a number of breakpoint options.
3. Tab to the following option near the bottom of the panel:

Statement no. of indirect command(s) to be executed: _____

and type the first indirect command statement number (leading zeroes are optional), as shown in the following screen.

```
CA InterTest MONITORING COMMAND BUILDER - COBOL BREAKPOINT LOCATIONS
SET breakpoint options for PROG=COBDEMO for location:

000479 Execute indirect command 100

After: _

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect: U080
Term ID (or .ANY) that will receive the breakpoints: U080
Statement no. of indirect command(s) to be executed: 10____
User ID (or .ANY) who will execute the program: .ANY
```

4. Press Enter to process this menu.

5. If you are setting a conditional breakpoint, enter the condition on the Conditional Breakpoint screen and press Enter.

Your source listing display shows the indirect command to be executed on the line above the statement with the U or C indicator.
To set a request breakpoint that executes an indirect command, follow the usual steps to set the request breakpoint from the menus. On the Request Breakpoint Selection menu, also type the statement number of the indirect command to execute, as shown in the following screen.

```
Replacement, Protection, and Special Options

CA InterTest MONITORING COMMAND BUILDER - REQUEST BREAKPOINT SELECTION 13

Set one or more types of Request Breakpoints in:
PROG=COBDEMO

ALL commands                    DL/I     DB2     CALLs
Address, Assign,                Storage Control  BMS
Handles, Push, Pop               Program Control  Trace Control
Terminal Control                 Interval Control  Dump Control
File Control                    Task Control     Batch Data Interchange
TD Control                      Journal Control  Built-In Functions
TS Control                      Syncpoints      Sys Prog Functions
Web access                      Business Trans  3270 Bridge

Enter 'n' to stop only every n'th time:
Term ID (or .ANY or .NO) where breakpoints will take effect:     
Term ID (or .ANY) that will receive the breakpoints:               
Statement no. of indirect command(s) to be executed:  0010
User ID (or .ANY) who will execute the program:                   

PF1 Help   2            3 End        4 Return     5            6
PF7        8            9           10           11           12

Review, Change or Delete Indirect Commands

To change, add, delete or review indirect commands directly from the Source Listing Display screen, type icmds in the command line or 11 in the Option # field and press Enter. CA InterTest displays the indirect commands for the default terminal, which is either the terminal you are working at or the terminal specified on the source listing profile.

For more information on the Indirect Commands facility, see the User Guide.

Replacement, Protection, and Special Options

CA InterTest has many options for specialized testing needs. Just a few are explained here; for more information on all the monitoring options, see the User Guide.

You can set or remove options from the Monitoring Command Builder menus. For more information, see Set Options from the Monitoring Menus (see page 63).
Replacement Options

Replacement options allow you to dynamically change the names of CICS resources such as programs, files, transient data queues, and temporary storage.

One reason to use replacement options is to correct errors so you can continue testing without recompiling. For example, suppose CA InterTest halts your program at an automatic breakpoint to prevent an AEIL abend caused by an incorrect file name. You can use the Replace File Name option to dynamically change the file name so testing can continue.

Another reason for using replacement options is to use different resources during testing than those hard-coded in the program. For example, you can specify that the program use a test file rather than the production file named in the program.

Protection Options

When CA InterTest monitors a program, it prevents the program from modifying storage areas it does not own and issuing non-CICS instructions. Protection options let you override the CA InterTest default rules for modifying main storage, the CSA, and load modules. You can do the following procedures with these options:

- Modify any area of main storage
- Modify areas in the CSA or CWA
- Modify a load module

These options also prevent a program from modifying specific areas of storage not normally protected.

The protection options, which are password protected, provide extra flexibility for special testing needs.

Special Options

Special options alter the CA InterTest standard monitoring procedures and help you design customized testing scenarios.

The No File Updating option prevents a monitored program from updating files. This option is in testing because your file data is unchanged at the end of program execution. This means:

- You can repeatedly test a program without restoring your files
- Programs can share the same file without interfering with each other’s work
- A test program can use a production file without affecting data integrity
Other options provide additional functions. For example:

- The FOL option lets CA InterTest continue monitoring even after a program branches directly to another program (wild branch).
- The MXS option limits the amount of CICS storage a program can use.
- The MXR option limits the number of CICS requests a program can issue.

Composite Module Testing

Composite support lets you take of all CA InterTest functions when you test composite modules. A composite module consists of separately compiled modules link-edited in one load module. These modules may be written in different languages and compiled at different times. Composite support lets you test and debug a called subroutine as if it were a separate program with full symbolic support.

Example

Suppose a COBOL program has several subroutines, some written in COBOL, some written in Assembler and PL/I. You can test and debug any of the subroutines just as if it were a main program by setting breakpoints, setting monitoring options, displaying the source listing, and displaying and modifying main and auxiliary storage.

Composite module monitoring is initiated by specifying the COMPOSITE command in the Source Listing Display screen before monitoring is activated (PF5) for the program.

```
CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING DISPLAY
COMMAND ===> COMPOSITE
Program= COBDEML Option #       Stmt #                             Margin= 01
Search=

00004 DATA DIVISION.
     00005 WORKING-STORAGE SECTION.
     00006  77 NUM-CHOICES PIC 9(4) COMP VALUE +2.
     00007  01 DFHAID COPY DFHAID.
00008*
     00009* COPYRIGHT = 5655-018 (C) COPYRIGHT IBM CORP. 1985 =
     00010* THIS MODULE IS "RESTRICTED MATERIALS OF=IBM"
     00011* LICENSED MATERIALS - PROPERTY OF IBM =
     00012* REFER TO COPYRIGHT INSTRUCTIONS =
     00013* FORM NUMBER G120-2083 =
     00014*
     00015  01 DFHAID.
     00016  02 DFHNUL PIC X VALUE IS ' '.
     00017  02 DFHENT PIC X VALUE IS '...'.
     00018  02 DFHCCLR PIC X VALUE IS ' '.
     00019  02 DFHCLR PIC X VALUE IS ' '.
```
When you specify the COMPOSITE command, CA InterTest displays the link-edit information for the main program and its subroutines. You can change this information online, as shown in the following screen:

Press PF5 to enable composite module monitoring so each program can be tested separately.

**Dump Analysis with CA SymDump**

CA SymDump for CICS—the CA InterTest companion symbolic dump facility for z/OS users—is a powerful online tool that automatically pinpoints the source statement responsible for an abend. CA SymDump was designed to complement CA InterTest, especially in production regions where programs are not usually monitored. Working with CA InterTest for CICS, CA SymDump for CICS lets you bring a dump back to life. Familiar CA InterTest source listing screens and other facilities, such as the ability to view and modify main and auxiliary storage, make it easy to resolve application dumps.

CA SymDump is option 5 Dump analysis on the CA InterTest Primary Option Menu.

With CA SymDump, you can do the following actions:

- Analyze dumps symbolically online, using CA InterTest source code listings
- Resolve CICS production dumps from any region; for example, debug production abends from your test region
- Manage your dump data set, selectively retaining the dumps you want to view and print, and discarding ones you do not need
When you use CA SymDump to view a dump, you can display the following items:

- CA InterTest breakpoint at the instruction that triggered the abend
- Formatted system areas
- Formatted trace table
- Registers and displacement at the abend

Example

Suppose a program not being monitored by CA InterTest for CICS abended with an ASRA. With CA SymDump for CICS, you can display the breakpoint at the statement that triggered the abend even though CA InterTest for CICS was not active in the region when the abend occurred.

Now you can use CA SymDump to solve the problem. In this example, you might want to display the contents of TASKNUM, or you might want to inspect a file or look at the formatted trace table. CA SymDump provides all the tools you need to diagnose and resolve the dump.

Because CA SymDump for CICS works just like CA InterTest for CICS, there is no learning curve. For more information, see the User Guide.
Chapter 5: Advanced Demo Session

The demo session in the second chapter detailed the basics of working with CA InterTest for CICS. Now we will look at some of the advanced features of CA InterTest for CICS.

**Note:** Only a handful of CA InterTest for CICS features are demonstrated in this demo session. For more information, see the User Guide or the Help facility.

If you are a new user, you might want to postpone this advanced demo session until you are more comfortable with the basic features of CA InterTest for CICS.

This chapter consists of the following independent sections. Each details a different advanced CA InterTest for CICS feature:

<table>
<thead>
<tr>
<th>Option Number</th>
<th>Demonstration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 01</td>
<td>Replace a file name.</td>
</tr>
<tr>
<td>Option 02</td>
<td>Limit the number of CICS requests issued by a program. Limit the amount of storage obtained by a program.</td>
</tr>
<tr>
<td>Option 03</td>
<td>Prevent a program from updating a file. Use request breakpoints and the FILE facility.</td>
</tr>
<tr>
<td>Option 04</td>
<td>Display variable length data.</td>
</tr>
<tr>
<td>Option 05</td>
<td>Display indexed COBOL table items.</td>
</tr>
<tr>
<td>Option 06</td>
<td>Detect and prevent a storage violation.</td>
</tr>
<tr>
<td>Option 07</td>
<td>Test a composite module.</td>
</tr>
</tbody>
</table>

Before proceeding with the demo session, you must complete the steps outlined in the next section.

This section contains the following topics:

- [Demo Preliminaries](#) (see page 90)
- [Execute the Demo Program](#) (see page 97)
- [Option 01: Replace a File Control ID](#) (see page 99)
- [Option 02: Limit CICS Storage and Requests](#) (see page 103)
- [Option 03: Prevent a Program from Updating a File](#) (see page 110)
- [Option 04: How to Display Variable Length Data](#) (see page 119)
- [Option 05: How to Work with Indexed Table Items](#) (see page 125)
- [Option 06: How to Detect a Storage Violation](#) (see page 133)
- [Option 07: How to Test a Composite Module](#) (see page 137)
Demo Preliminaries

The advanced demo session uses COBDEMO just as the basic session did. Before beginning, you must set some breakpoints that will be used in different sections of the advanced demo session.

Set Unconditional Breakpoints

To begin, you must bring up the Source Listing display for the COBOL demo program. This was covered in the basic demo session.

1. Press PF4 to display the Profile panel. Verify that the Display window is set to "N" and that the AutoKeep Display is "ON". If not, change either or both of them and press Enter.

2. Press PF3 to return to the listing.

Now you are going to set unconditional breakpoints at three procedure names. Setting unconditional breakpoints lets you halt the program so that you can set options and check the contents of data items.

The quickest way to set breakpoints at several procedure names is to display a list of all procedure names, and type a u next to the ones where you want breakpoints.
3. Type `l .px` in the command line or `7` in the Option # field, and press Enter.

![List of Procedure Names](image)

CA InterTest for CICS displays a list of the demo program’s procedure names, as shown in the following figure.

You are going to set unconditional breakpoints at three procedure names: AFTER-REWRITE, DO-READ-VAR, and LOOP-RTN.

4. Type a `u` to the left of AFTER-REWRITE, DO-READ-VAR, and LOOP-RTN.

5. Press Enter.

![Breakpoints Set](image)

CA InterTest for CICS sets the breakpoints and redisplays the screen with uppercase Us.
Set Request Breakpoints

Request breakpoints make it easy for you to halt your program at various points so you can check program processing. For example, you can halt the program before every READ command, single-step through the read instructions, and then check main storage to make sure the program has read the correct record.

You can set request breakpoints to halt a program before CICS commands and macros and other calls, such as calls to DL/I or DB2. You can set request breakpoints prior to all CICS commands (as you might have done with IBM's CEDF) or prior to specific commands, such as File Control or Program Control commands.

For option 03 of the advanced demo session, you need to set request breakpoints at all File Control REWRITE commands in the demo program. To do this, do the following:

1. Specify the RBP command on the command line.
   
   CA InterTest for CICS displays the Request Breakpoint Selection menu on which you specify the type of request breakpoint, such as: all CICS commands, File Control and Task Control.

<table>
<thead>
<tr>
<th>Set one or more types of Request Breakpoints in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROG=COBDEMO</td>
</tr>
<tr>
<td>ALL commands  _ DL/I  _ DB2  _ CALLs</td>
</tr>
<tr>
<td>Address, Assign, Handles, Push, Pop</td>
</tr>
<tr>
<td>Terminal Control</td>
</tr>
<tr>
<td>$ File Control</td>
</tr>
<tr>
<td>TD Control</td>
</tr>
<tr>
<td>TS Control</td>
</tr>
<tr>
<td>Web access</td>
</tr>
<tr>
<td>Storage Control</td>
</tr>
<tr>
<td>Program Control</td>
</tr>
<tr>
<td>Trace Control</td>
</tr>
<tr>
<td>Interval Control</td>
</tr>
<tr>
<td>Dump Control</td>
</tr>
<tr>
<td>Task Control</td>
</tr>
<tr>
<td>Batch Data Interchange</td>
</tr>
<tr>
<td>Journal Control</td>
</tr>
<tr>
<td>Built-In Functions</td>
</tr>
<tr>
<td>Syncpoints</td>
</tr>
<tr>
<td>Sys Prog Functions</td>
</tr>
<tr>
<td>Business Trans</td>
</tr>
<tr>
<td>Business Trans</td>
</tr>
<tr>
<td>3270 Bridge</td>
</tr>
</tbody>
</table>

   Enter 'n' to stop only every n'th time:       ____
   Term ID (or .ANY or .NO) where breakpoints will take effect: ___
   Term ID (or .ANY) that will receive the breakpoints: ___
   Statement no. of indirect command(s) to be executed: ___
   User ID (or .ANY) who will execute the program:    ___

2. Tab to the File Control field and type an $.

3. Press Enter.
CA InterTest for CICS displays the File Control Request Breakpoint Selection menu shown in the following screen.

![CA InterTest Monitoring Command Builder - Request Breakpoint Selection](image)

On this menu you can select the specific File Control commands where you want to set request breakpoints. In this case select REWRITE to halt the program prior to each File Control REWRITE command.

4. Tab to the REWRITE command field.
5. Type an s and press Enter.

CA InterTest for CICS sets request breakpoints prior to each File Control REWRITE command in the demo program and redisplays the Program Monitoring Menu.
The next topic is necessary only if you want to use the composite support option. Composite support is a CA InterTest for CICS feature designed to aid programmers who are responsible for testing and debugging called subroutines.

- If you do not want to use the composite support option, skip to Execute the Demo Program (see page 97).
- If you do want to use the composite support option, perform the steps in Composite Support that follows.

**Composite Support**

Composite support lets you use CA InterTest for CICS to test and debug composite modules. A *composite module* is a load module, defined to CICS as a program, which consists of separately compiled or assembled parts brought together when the module is link-edited. In the demo session, the part of the composite module that receives control from CICS is referred to as the *main program*; the remaining programs are referred to as *subroutines*. The main program and subroutines can be written in the same or different languages.

Composite support lets you test and debug a subroutine, such as an external procedure, as if it were a separate program with full symbolic support. This means you can set breakpoints and other monitoring options individually for any subroutine. CA InterTest for CICS generates automatic breakpoints in the subroutine when it detects an error that would otherwise cause the program to abend.

To demonstrate this feature, you set composite support for a composite module, to which the demo program links. The composite module that you use is COBDEML.

The composite module has two called subroutines that you want CA InterTest for CICS to monitor—subroutine CSBIN25 written in COBOL, and subroutine ASBIN25 written in Assembler.

**Follow these steps:**

1. In the Program Monitoring Menu, press PF4 until the Source Listing Display is redisplayed.
2. In the Program field, overtype the name of the demo program with the name of the composite module COBDEML, and press Enter.
CA InterTest for CICS displays the source listing for the composite module.

3. Type COMPOSITE on the command line and press Enter.

4. Press PF5 to activate composite program monitoring for COBDEML.
Check the Monitoring Status Display

Before continuing, we will check the Monitoring Status display for the program COBDEMO. You should see the options you set for the program, which are needed to complete the Demo Options in the rest of this guide.

1. On the Program Monitoring menu, scroll the Options list to the beginning. Either enter the TOP command and press Enter or press PF7.

```
------------------- CA InterTest V9.1 PROGRAM MONITORING -------------------

COMMAND =>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Program name (or .ALL, .OPTIONS or generic)</td>
</tr>
<tr>
<td></td>
<td>User (or .ANY) for whom the program is monitored</td>
</tr>
<tr>
<td>s</td>
<td>Display and/or remove monitoring options (S only)</td>
</tr>
<tr>
<td>Monitor</td>
<td>Monitoring (R removes monitoring and all options previously set)</td>
</tr>
<tr>
<td>UBP</td>
<td>Unconditional breakpoints (specific program only)</td>
</tr>
<tr>
<td>CBP</td>
<td>Conditional breakpoints (specific program only)</td>
</tr>
<tr>
<td>RBP</td>
<td>Breakpoints for CICS, DB2, DL/I or external CALL requests</td>
</tr>
<tr>
<td>Stmt Trace</td>
<td>Statement tracing and data monitoring (COBOL only)</td>
</tr>
<tr>
<td>New copy</td>
<td>Fetch new copy of program and reset monitoring options (S only)</td>
</tr>
<tr>
<td>Commands</td>
<td>Indirect commands defined for a specific COBOL or PL/1 program</td>
</tr>
<tr>
<td>Replace</td>
<td>CICS resource name replacement options</td>
</tr>
<tr>
<td>Protect</td>
<td>Storage protection monitoring options</td>
</tr>
<tr>
<td>Special</td>
<td>Other options (storage allocation, file updating, etc.)</td>
</tr>
<tr>
<td>Composite</td>
<td>Monitor multi-CSECT program’s separately compiled components</td>
</tr>
</tbody>
</table>

PF1 Help 2 3 End 4 Return 5 6
PF7 Backward 8 Forward 9 10 11 12
```

2. Type an s next to the Status option to obtain a Monitoring Status display of the COBDEMO program.
The Monitoring Status display for COBDEMO should show the three breakpoints needed to perform the Demo Session Options; the Request Breakpoint (RBP) at all REWRITE commands, and the two Unconditional Breakpoints (UBPs).

```
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBDEMO</td>
<td>Program monitor entry</td>
<td>IBMCOB 3.2</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>ANY</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Option ID</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>From, to terminals</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Option ID</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>From, to terminals</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Option ID</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>From, to terminals</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Option ID</td>
</tr>
</tbody>
</table>
```

3. Press PF3 to exit back to the Program Monitoring menu.

4. Type =x in the Command field to exit the menus, then PF3 to exit Source Listing.

Exiting the menus and Source Listing returns you to CICS, where you can begin executing the COBDEMO program.

**Execute the Demo Program**

Now you are ready to execute the demo program.

1. Exit to CICS using PF3 End. You may need to press PF3 a number of times.

2. On a clear CICS screen, enter the transaction identifier DEMC of the COBDEMO program.

3. Press Enter.
The following Welcome Screen appears.

```
**********************************************************************
*****                             Welcome to the                         *****
*****                       CA InterTest Demo Session                    *****
*****                                                                    *****
*****              Before proceeding, please have on hand the            *****
*****              guide which accompanies the Demo Session.             *****
*****                                                                    *****
*****     Please make sure that the program COBDEMO is monitored by      *****
*****     CA InterTest. This program will abend if it is not monitored.  *****
*****     To turn the monitor on, press CLEAR and follow the steps       *****
*****     outlined in the documentation.                                *****
*****     If the monitor is already on, press ENTER to begin the        *****
*****     Basic Demo Session or PF2 to go to the Options Menu.          *****
**********************************************************************
```

From the Welcome Screen you can access the Options Menu. Each of the options is described in detail in the sections that follow.

4. Press PF2 to go to the Options Menu of the demo.

The demo program displays the Options Menu, as shown next.

```
**********************************************************************
*****                             CA InterTest Demo Session                         *****
*****                             Options Menu                                     *****
**********************************************************************

01  Replace file control ID
02  Limit CICS storage and requests
03  No file updating
04  Display variable length data
05  Work with table items
06  Storage violation detection
07  Composite support

Key in request:  01

Press ENTER to continue or CLEAR to terminate.
```
Option 01: Replace a File Control ID

This section of the demo session illustrates how to replace a File Control ID. The demo program is attempting to read and display a record, but the file name specified in the program is incorrect, possibly because it was misspelled.

**Note:** Before proceeding, ensure you have completed the steps outlined in the Demo Preliminaries section, and that you have the CA InterTest Demo Session Options Menu displayed on your screen.

1. Select option 01 and press Enter.
   
   The demo program displays the following screen, which describes what occurs in this part of the demo session.

   ```plaintext
   ******************************************************
   ****                     ****
   ****                     ****
   ****   CA InterTest Demo Session                ****
   **** Replace File Control ID Option              ****
   ****                     ****
   ******************************************************
   The program attempts to read file PROTH. Because the file name has been incorrectly specified, this would cause a DSIDERR condition which would result in an AEIL abend. Instead, here is what will happen:
   1. CA InterTest halts the program at an automatic breakpoint.
   2. You set the Replace File Control option to change the file name without recompiling.
   3. You re-execute the program from the point at which it first tried to read the file.

   When this section of the demo completes, you will return to the Options Menu.

   Press ENTER to continue or CLEAR to terminate.
   ```

2. Press Enter.
   
   CA InterTest for CICS halts the demo program at an automatic breakpoint.
Prevent an AEIL Abend

The following screen shows that CA InterTest for CICS has halted the demo program at an automatic breakpoint before an AEIL abend could occur.

![Screen shot showing CA InterTest for CICS halted demo program]

The abend that CA InterTest for CICS prevented would have resulted from the EXEC CICS STARTBR command. CA InterTest for CICS highlighted the CALL statement that is part of the generated EXEC CICS STARTBR command.

**What is the problem?**

The data set name PROTH is wrong; it should be PROTHLF.

**How can it be fixed?**

CA InterTest for CICS lets you dynamically correct the file name and continue testing. The following section explains how this is done.

**Change the File Name**

CA InterTest for CICS lets you dynamically change an incorrect file name in the middle of your test session. This feature lets you continue testing without recompiling your program. In this case, you can correct a simple typo in the demo program without interrupting your test session or changing your File Control Table.
To change the file name, you need to access the Replacement Options menu.

1. Specify the RO command on the Source Listing Display command line.
   CA InterTest displays the Replacement Options menu.
   The Set Replacement Options menu lets you dynamically change a program name,
   file name, transient data queue name, or temporary storage ID.

2. Tab to the Replace file name field.

3. Specify the incorrect file name (proth) in the first column and the correct file name
   (prothlf) in the second column, and press Enter.

   CA InterTest dynamically replaces the incorrect file name when the program
   executes. CA InterTest then redisplays the Program Monitoring menu.

4. Press PF4 until the Source Listing Breakpoint Screen is redisplayed.

**Resume Execution**

Now that you have dynamically corrected the file name, you are ready to resume
program execution. However, be aware that when an error occurs in a paragraph
containing a CICS command, you should back up to the paragraph-name rather than
resume execution at the specific instruction (some CICS internal processing might have
occurred prior to the breakpoint). In this case it is important to resume execution at
READ-DATASET to ensure that the demo proceeds as expected.

1. On the source listing breakpoint screen, position the listing so the READ-DATASET
   label is displayed by pressing PF7, tab to READ-DATASET, and enter g (go).
2. Press Enter to resume program execution.

The program resumes execution and displays the record it has read on your screen, as illustrated in the following example. The demo program succeeded in reading a record from file PROTHLF (the CA InterTest for CICS Help file). The demo program displays the first record in that file.
Review What Happened

In this part of the demo session the following occurred:

- The demo program was stopped at an automatic breakpoint caused by an incorrect file name.
- You dynamically corrected the file name and resumed program execution.

Replacing the incorrect file name in the demo program allowed you to continue testing that program without recompiling it. Of course, if one of your programs or the CICS FCT has an incorrect file name, you would have to correct the source code.

In this case the error was caused by a simple typo. However, the Replacement Options are useful in many situations. For example:

- You can dynamically change a file name to test a program using a different file. This lets you use a test file without altering the name of the production file hard-coded in the program.
- You can change a program name or transient data queue name to test a program or queue other than the ones that are hard-coded.

Note: This concludes Option 01 of the advanced demo session.

1. Press Enter to continue the test session.
   or
   Press Clear to end the session.

Option 02: Limit CICS Storage and Requests

This section of the demo session shows how to limit CICS main storage and CICS requests for a specific program (the demo program).

Two CA InterTest for CICS options are detailed in this part of the demo session:

- The MXS option limits the amount of main storage obtained by a task. This option is useful in detecting program errors that cause a task to acquire an excessive amount of storage.
- The MXR option limits the number of CICS requests, commands and macros that a program can issue. This option is useful in detecting program loops that generate an excessive number of CICS requests.

Note: Before proceeding, be sure you have completed the steps outlined in the Demo Preliminaries section, and that you have the CA InterTest Demo Session Options Menu displayed on your screen.
1. Select option 02, and press Enter.

The demo program displays a screen that describes what occurs in this part of the demo session.

```
**********************************************************************
****                                                              ****
****                  CA InterTest Demo Session                   ****
****          MXR Option - Limit Number of CICS Requests          ****
****                  MXS Option - Limit Amount of CICS Storage     ****
****                                                              ****
**********************************************************************
```

The program has a loop containing a CICS request and another loop containing a GETMAIN request. Here is what will happen:

1. The program is halted at an unconditional breakpoint before the loops.
2. You set the MXS and MXR options to limit storage and CICS requests.
3. The program continues to execute.
4. CA InterTest halts the program at automatic breakpoints when the limits are exceeded.
5. You remove the options and the program completes execution.

When this section of the demo completes, you will return to the Options Menu.
Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

CA InterTest for CICS halts the program at the procedure name LOOP-RTN (because you set an unconditional breakpoint there as part of the Demo Preliminaries). Because a procedure name does not have executable code, the breakpoint you set halts execution before the first verb in the paragraph.

In this case, CA InterTest for CICS halts the demo program at the highlighted PERFORM command. Now you can access the Special Options menu to set the options you need for testing.

**Set the MXS and MXR Options**

1. In the Command field, specify the SO command.

CA InterTest for CICS displays the Special Options menu. The Special Options menu lets you set several options to change how CA InterTest for CICS monitors a program.

You are going to set two options, as shown following:

- MXS, to limit the amount of CICS main storage acquired by the demo program
- MXR, to limit the total number of CICS requests issued by the demo program
2. Specify 50,000 bytes as the maximum amount of main storage the demo program obtains, and specify 20 as the maximum number of CICS requests the demo program issues.

3. Tab to the MXS field and type in 60000, as shown in the following screen.

4. Tab to the MXR field and type in 20, as shown in the following screen.

5. Press Enter.

```
CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS

SET one or more options to override the default monitoring rules in:
PROG=COBDEMO

Enter X next to each option desired:

Source Listing Breakpoint (SLB) _
No file updating (NUP) _
Reentrancy check (RNT) _

Follow monitoring (ON, name, NOPPT) (FOL) ______
Number of times to be monitored (MUS) ______
Limit total size of CICS storage (MXS) 60000 _
Limit total number of CICS requests (MXR) 20 ______
Structure Display Format (HEX,DATA) (SDF) _____

Set local automatic breakpoint ('*', TERMID, .ANY, OFF) ______
Limit monitoring to your TERMINAL - '*' or TERMID: ______
User ID (or .ANY) who will execute the program: .ANY
```

CA InterTest for CICS limits main storage and CICS requests for the demo program and redisplays the Program Monitoring screen.

6. Press PF4 until you return to the Source Listing Breakpoint screen.

7. Press PF5 to continue program execution.

The demo program resumes execution.

**Limit CICS Requests**

The next screen you see is an automatic breakpoint display screen. CA InterTest for CICS has halted the demo program at an automatic breakpoint because the maximum number of CICS requests (20) has been exceeded.
Note: If you are working on a system with LE 370 runtimes, you might reach the automatic breakpoint for the MXS (max storage size) first. If you do, simply perform the demo in the following order:

1. Remove the MXS options by typing STATUS, in the command line then 'R' next to the MXS option, and hit enter. Resume execution.
2. After stopping for the MXR trigger value, remove it the same as the MXS and continue.
3. The demo will execute to completion.

The instruction that triggered the automatic breakpoint is within a loop containing a CICS request.

```
CA InterTest for CICS V9.1 - PROTSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ===>
Program= COBDEMO Option # Stmt # Margin= 01
Search=
-------------------------------------------------------------------
000001 EXEC CICS ASKTIME
000002              END-EXEC.
A   ==>     Call 'DFHEI1' using by content x'100200000700001300f0f0f4f5f4'
     ==>     MXR (max CICS requests) trigger value exceeded.
     ==>     Press PF1 for a detailed description.
     ==>     000004 '404040' end-call.
     000005
     000006 MXS-OPTION.
     
```
Remove the MXR Option

Now you can remove the MXR option so that the demo program continues executing. If this were your own program, you would examine the instructions that are generating excessive CICS requests.

1. Type **status** on the command line and press Enter.
   
   CA InterTest for CICS displays the Monitoring Status screen.

   ![Monitoring Status Screen](image)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBDEMO</td>
<td>Program monitor entry</td>
<td>IBMCOB 3.2</td>
</tr>
<tr>
<td></td>
<td>Waiting at breakpoint</td>
<td>Task 00156, ABP since 10:33 a.m.</td>
</tr>
<tr>
<td>-</td>
<td>User monitoring options</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td>Symbolic listing file</td>
<td>PROTSYM</td>
</tr>
<tr>
<td>-</td>
<td>MXR Maximum CICS requests</td>
<td>20</td>
</tr>
<tr>
<td>-</td>
<td>MXS Maximum total storage</td>
<td>58000</td>
</tr>
<tr>
<td>-</td>
<td>RBP Request breakpoint(s)</td>
<td>REWRITE</td>
</tr>
<tr>
<td></td>
<td>Option ID</td>
<td>ECEAD6CB</td>
</tr>
<tr>
<td></td>
<td>From, to terminals</td>
<td>U061, U061</td>
</tr>
<tr>
<td>-</td>
<td>UBP Unconditional breakpoint</td>
<td>'AFTER-REWRITE'</td>
</tr>
<tr>
<td></td>
<td>Option ID</td>
<td>D87A3068</td>
</tr>
<tr>
<td></td>
<td>From, to terminals</td>
<td>U061, U061</td>
</tr>
<tr>
<td>-</td>
<td>UBP Unconditional breakpoint</td>
<td>'LOOP-RTN'</td>
</tr>
<tr>
<td></td>
<td>Option ID</td>
<td>6CD871EC</td>
</tr>
</tbody>
</table>

2. Remove the MXR option by typing an **r** next to the MXR breakpoint's entry and pressing Enter.
   
   CA InterTest for CICS removes the breakpoint.

3. Press PF3 to return to the Source Listing Breakpoint screen.

4. Press PF5 to resume program execution.
Limit Acquisition of Storage

The next screen you see is an automatic breakpoint display. CA InterTest for CICS halts the demo program at an automatic breakpoint because the maximum main storage limit of 60,000 has been exceeded. In most cases, the instruction that triggered the automatic breakpoint is within a loop, which contains a GETMAIN request.

```
CA InterTest for CICS V9.1 - PROSYM FILE ABEND DETECTED BREAKPOINT
COMMAND ===> Program= COBDEMO Option # Stmt # Margin= 01

<table>
<thead>
<tr>
<th>________ DFHB0020</th>
<th>+06144.</th>
</tr>
</thead>
<tbody>
<tr>
<td>000825* END-EXEC.</td>
<td></td>
</tr>
<tr>
<td>000826 Move 6144 to dfhb0020</td>
<td></td>
</tr>
<tr>
<td>=&gt; Call 'DFHEII' using by content x'0c02c00007000080f0f0f4f6f8</td>
<td></td>
</tr>
<tr>
<td>=&gt; MVS (max storage size) trigger value exceeded.</td>
<td></td>
</tr>
<tr>
<td>=&gt; Press PF1 for a detailed description.</td>
<td></td>
</tr>
<tr>
<td>=&gt; '404040' by reference ADDRESS OF GETMAIN-AREA by reference</td>
<td></td>
</tr>
<tr>
<td>000829 dfhb0020 end-call.</td>
<td></td>
</tr>
<tr>
<td>000830 NO-STORAGE.</td>
<td></td>
</tr>
<tr>
<td>000831 MOVE 'B' TO TASK-SWITCH3.</td>
<td></td>
</tr>
<tr>
<td>000832 MOVE 'NO STORAGE' TO ERRORO.</td>
<td></td>
</tr>
<tr>
<td>000833*EXEC CICS SEND</td>
<td></td>
</tr>
<tr>
<td>000834* MAP ('DERROR')</td>
<td></td>
</tr>
<tr>
<td>000835* MAPSET ('IN25CMP')</td>
<td></td>
</tr>
<tr>
<td>000836* ERASE</td>
<td></td>
</tr>
</tbody>
</table>
```
Remove the MXS Option

Now you can remove the MXS option so that the demo program continues executing. If this were your own program, you would examine the instructions that are causing the program to obtain excessive amounts of storage.

1. Type `status` on the command line and press Enter.
   
   CA InterTest for CICS displays the Monitoring Status screen.

   ![Monitoring Status Screen]

   2. Remove the MXS option by typing an `r` next to the MXS breakpoint's entry and pressing Enter.
      
      CA InterTest for CICS removes the breakpoint.

   3. Press PF3 to return to the Source Listing Breakpoint screen.

Demo Program Completes Execution

1. Press PF5 to resume program execution.
   
   The demo program completes execution. CA InterTest for CICS returns to the demo's Options Menu.
Option 03: Prevent a Program from Updating a File

Review What Happened

In this part of the test session you set two options to help test the demo program, as shown following:

- The MXR option lets you limit the number of CICS requests the program could issue. When the demo program exceeded this limit, CA InterTest for CICS halted the program. This feature helps you find program instructions within a loop that generate an excessive number of CICS requests.

- The MXS option lets you limit the amount of main storage used by the program. When the demo program exceeded the amount of storage you specified, CA InterTest for CICS halted the program. This feature helps you find commands or macros within a loop that are obtaining excessive amounts of storage.

You have now completed Option 02 of the advanced demo session.

1. Press Enter to continue with the demo session.
   or
   Press Clear to terminate the session.

Option 03: Prevent a Program from Updating a File

This section of the demo session details how to execute a program without having it update a file.

The No File Updating option is useful when you are testing a program. Preventing a program from updating a file means your test data remains unchanged at the end of each program execution, so you can test the program repeatedly without having to recreate test data.

Many programs can use the same test file without interfering with each other’s work. You can even allow a test program to use a production file because the integrity of the file is preserved.

Note: Before proceeding, be sure you have completed the steps outlined in Demo Preliminaries and that CA InterTest Demo Session Options Menu is displayed on your screen.
Also be sure that the CA InterTest for CICS checkpoint file (PROTCPF) has been defined to your CICS region and that the file's current status is Open and Enabled.

1. Select option 03, and press Enter.

   The demo program displays the following screen that describes what occurs in this part of the demo session.

```
**********************************************************************
****                                                              ****
****                      CA InterTest Demo Session                   ****
****                      No File Updating Option                    ****
****                                                              ****
**********************************************************************

The program reads a record and changes data in the record. Here is what will happen:

1. CA InterTest halts the program at a request breakpoint.
2. You set the No File Updating option to prevent COBDEMO from updating the file.
3. When COBDEMO rewriting the file, the file is not updated.
4. After the rewrite you use the FILE facility to confirm that the file has not been updated.

When this section of the demo completes, you will return to the Options Menu.

   Press ENTER to continue or CLEAR to terminate.
```

2. Press Enter.

   CA InterTest for CICS displays a request breakpoint screen.
Halt the Demo Program at a Request Breakpoint

CA InterTest for CICS has halted the demo program prior to the first File Control REWRITE command as you specified during the Demo Preliminaries section. The highlighted CALL instruction is part of the generated EXEC CICS REWRITE command.
Setting request breakpoints lets you inspect the values of program variables and set options prior to all or specified CICS commands and macros and other program calls.

1. Scroll backward to look at the EXEC CICS REWRITE command.

CA InterTest for CICS displays the following screen.

```
000902 GO TO SEND-REWRITE-RETURN.
000903 NUP-READ.
000904 PERFORM SET-UP-READ.
000905+EXEC CICS READ DATASET(TASK-PROTCPF) INTO(VSAM-AREA) EQUAL
000906+ LENGTH(REC-LEN) RIDFLD(REC-RBA) UPDATE
000907* END-EXEC.
000908 Call 'DFHEI1' using by content x'0602f0000700008400f0f0f5f1f9
000909- '404040' by reference TASK-PROTCPF by reference VSAM-AREA by
000910- reference REC-LEN by reference REC-RBA end-call.
000911 VSAM-REWRITE.
000912 MOVE 'THIS IS NOT A NAME ' TO VSAM-NAME.
000913+EXEC CICS REWRITE
000914+ DATASET(TASK-PROTCPF)
000915+ FROM(VSAM-AREA)
000916+ LENGTH(REC-LEN)
```

According to the EXEC CICS REWRITE command, the demo program will update the file PROTCPF (specified in the DATASET parameter of the EXEC CICS REWRITE command).

**Note:** If you do not have the AutoKeep Display facility running, to view the contents of DATASET(TASK-PROTCPF), you can enter Search=TASK-PROTCPF to position the listing where it is defined, and add it to the Keep window (type k next to the line defining TASK-PROTCPF, position the cursor in the field name, and press Enter).

However, you are going to set the No File Updating option to prevent the demo program from updating the file.

To do so, access the Special Options menu.

### Set the No File Updating Option

1. As we did when we set and removed the MXR and MXS options, bring up the Program Monitoring screen by specifying the SO option on the command line.

   CA InterTest for CICS displays the Special Options menu.

2. Tab to the NUP field.
3. Type an x, and press Enter.

<table>
<thead>
<tr>
<th>CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET one or more options to override the default monitoring rules in:</td>
</tr>
<tr>
<td>Enter X next to each option desired:</td>
</tr>
<tr>
<td>Source Listing Breakpoint (SLB) _</td>
</tr>
<tr>
<td>Reentrancy check (RNT) _</td>
</tr>
<tr>
<td>Follow monitoring (ON, name, NOPPT) (FOL) _</td>
</tr>
<tr>
<td>Limit total size of CICS storage (MXS) _</td>
</tr>
<tr>
<td>Set local automatic breakpoint ('*', TERMID, .ANY, OFF) ___</td>
</tr>
<tr>
<td>User ID (or .ANY) who will execute the program: ___</td>
</tr>
</tbody>
</table>

CA InterTest for CICS sets this option for the demo program.

4. Press PF4 until you return to the Source Listing Breakpoint screen.

Before resuming program execution, look at the MOVE instruction just before the EXEC CICS REWRITE command. It specifies that the data string THIS IS NOT A NAME be moved to the data field VSAM-NAME. If the demo program actually updates the file, that data string displays at the beginning of the first record in that file.

5. Press PF5 to continue program execution.

The demo program resumes execution.
Use FILE to Inspect the Record

CA InterTest for CICS now halts the demo program at an unconditional breakpoint you set at procedure name AFTER-REWRITE in the Demo Preliminaries section.

Because a procedure name does not have executable code, the unconditional breakpoint halts execution before the first verb—in this case, the highlighted MOVE command.

You set the breakpoint at the paragraph immediately following the REWRITE command so you could use the CA InterTest for CICS FILE facility to confirm that the demo program did not, in fact, update the file.

   CA InterTest for CICS displays the Breakpoint Primary Option Menu.
2. Select option 1 Main Menu.
   CA InterTest displays the Primary Option Menu.
3. Select 4 Auxiliary storage.

CA InterTest for CICS displays the Auxiliary Storage Menu.

```
CA InterTest for CICS V9.1 AUXILIARY STORAGE MENU

OPTION => 1

Select an auxiliary storage type, specifying optional criteria below.

1  Files  - Display/select files for access
2  DB2 database - Invoke DB2 SQL interface facility
3  DL/I database - Access DL/I database
4  TD queues - Display/select transient data queues for access
5  TS queues - Display/select temporary storage queues for access

Type specific or generic file/queue name(s):
(Valid mask characters are * and/or +)

protcpf _______________________________

.
```

4. Type a 1 in the Option field and protcpf for the file name.

5. Press Enter.

CA InterTest for CICS displays the File Selection menu.

6. Type an s next to the file name and press Enter.

CA InterTest for CICS displays the initial File Facility screen.

```
DATATYPE= FC FILEID= PROTCPF MODE= LOG=ON TDDEST= PASSWORD=
FUNC= SUBFUNC= RETMETH= ARGTYP= SRCHTYP=
MESSAGE=
RETNRCID= RCID=
DATA= SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF *0123456789ABCDEF*
LOC 0000 ........ ........ ........ ........ ........

CA InterTest for CICS V9.1

Copyright © 2014 CA. All rights reserved
```

CA InterTest for CICS displays the first record in the PROTCPF file.

```
DATATYPE= FC FILEID= PROTCPF MODE=BROWSELOG=ON TODEST= PASSWORD=
FUNC= NEXT SUBFUNC= RETMETH= ARGTYP= SRCHTYP=
MESSAGE= CAIN0601 RECORD OBTAINED FOR VIEWING
RETNRCID=404040404040405C 5C5C4040 40404040404040
RCID=
DATA= SIZE=02A9
FORMAT= D 00112233 44556677 8899AAAABB CCDEEFF
LOC 0000 40404040 40404040 40404040 40404040
0000 40404040 40404040 40404040 40404040 40404040 40404040
0010 D4C540E3 C8C540C3 C8C5C50C 40D7D9
0020 D4C540E3 C8C540C3 C8C5C50C 40D7D9
0030 4040405C 5C5C4040 40404040 40404040 40404040
0040 40404040 40404040 40404040 40404040 40404040
0050 00000000 00000000 00000000 00000000 00000000 00000000
0060 00000000 00000000 00000000 00000000 00000000 00000000
0070 00000000 00000000 00000000 00000000 00000000 00000000
0080 00000000 00000000 00000000 00000000 00000000 00000000
0090 00000000 00000000 00000000 00000000 00000000 00000000
00A0 00000000 00000000 00000000 00000000 00000000 00000000
00B0 00000000 00000000 00000000 00000000 00000000 00000000
00C0 00000000 00000000 00000000 00000000 00000000 00000000
00D0 00000000 00000000 00000000 00000000 00000000 00000000
00E0 00000000 00000000 00000000 00000000 00000000 00000000
00F0 00000000 00000000 00000000 00000000 00000000 00000000
1 Help 2 Format C 3 End 4 ENDB 5 PREV 6 DataType DL
7 Page bwd 8 Page fwd 9 Caps Off 10 Top 11 Bottom 12
```

Note: Your version of this screen might differ slightly.

The previous screen displays the contents of the first record in both hexadecimal and character formats. It is clear from looking at the character display that the data string THIS IS NOT A NAME has not been moved to the beginning of this record.

Note: Setting the No File Updating option prevented the demo program from updating this file, without affecting the demo program's source code.

Before returning to the demo program, we will discuss some of FILE's capabilities.

**A Brief Look at FILE's Capabilities**

The FILE facility lets you view, update, add and delete records and search for character strings. You can perform these functions at any time (for example, while your program is at a breakpoint). You can also use FILE when no program is executing to perform routine file maintenance.

FILE displays records in dump format, as shown in the previous screen, character format, vertical format, and structured format. To display the record in different formats, press PF2. For structured format, which displays records or DL/I segments on a field-by-field basis, you also must identify the program containing the structure.

You can use FILE with VSAM and BDAM files, DL/I and DB2 databases, temporary storage records, and transient data records.
Take advantage of the FILE facility when testing your own programs. Use FILE to change your test records or to create additional records in the middle of a test session. You can also use FILE to make sure your program has successfully updated a file.

Now we will return to testing the demo program.

**Demo Program Completes Execution**

1. Press Clear.
   
   CA InterTest for CICS redisplay the File Selection screen.
2. Press PF4 until you return to the Source Listing Breakpoint screen.
3. Press PF5 to resume program execution.
   
   You return to the demo program's Options Menu.

**Review What Happened**

In this part of the demo session you took advantage of several CA InterTest for CICS features. You were able to do the following:

- Use a request breakpoint to halt the demo program prior to the first File Control REWRITE command.
  
  Request breakpoints make it easy for you to halt your program at various points, such as before all HANDLE CONDITION commands or all Terminal Control commands. When your program is halted, you can inspect main storage or auxiliary storage to detect errors and review program logic.

- Use the No File Updating option to prevent a program from updating a file.
  
  Preventing a program from updating a file means you can test a program repeatedly without having to recreate test data. Many programs can share the same test file because no program will actually change it. And, a test program can even use a production file without corrupting it.

- Display a record in a file.
  
  Displaying a record in a file lets you see whether your program is working correctly. The FILE facility also lets you add, delete, and update records to meet your individual testing needs and maintain files without special one-time programs.

  **Note:** This concludes Option 03 of the advanced demo session.

1. Press Enter to continue with the demo session.
   
   or
   
   Press Clear to terminate the session.
Option 04: How to Display Variable Length Data

This section of the demo session details how you can see the current length of variable length records.

Note: Before you begin this section, complete the steps outlined in Demo Preliminaries (unless you did so earlier in this session). The CA InterTest Demo Session Options Menu should be displayed.

1. Select option 04, and press Enter.

The demo program displays a screen that describes what occurs in this part of the demo session.

```
**** CA InterTest Demo Session ****
**** Display Variable Length Data ****
**********************************************************************
This feature lets you see the current length and contents of variable length data and records at any point in your program. Here is what will happen:

1. CA InterTest halts COBDEMO at an unconditional breakpoint.
2. You use the CORE facility to see the original record.
3. COBDEMO continues to execute and changes the length of the record.
4. You use CORE again to see the updated record.

When this section of the demo completes, you will return to the Options Menu.

Press ENTER to continue or CLEAR to terminate.
```
2. Press Enter.

CA InterTest for CICS halts the execution of the demo program at an unconditional breakpoint set during the Demo Preliminaries section.

The unconditional breakpoint was actually set at procedure name DO-READ-VAR. Because a procedure name does not have executable code, the unconditional breakpoint halts execution before the first verb in the paragraph. In this case, the first verb is the highlighted MOVE command below the comment lines. Halting execution at this point in the program lets you see the contents of a variable length record before the demo program changes its length.

Execution of the highlighted MOVE statement changes the length of the 01-level record VARIABLE-LENGTH-RECORD. Its definition tells us why.

3. Type `L variable-length-record` into the command line, and press Enter.
CA InterTest for CICS finds the definition of VARIABLE-LENGTH-RECORD in Working-Storage and displays that portion of the compile listing.

```
CA InterTest for CICS V9.1 - PROSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ===> Program= COBDEMO Option # Stmt # Margin= 01
Search= +00100.

____ VAR-REC-LEN | +00100.
  000097 OCCURS 5 TIMES
  000098 INDEXED BY DISTRICT-X.
  000099 01 VARIABLE-LENGTH-RECORD.
    000100 03 VAR-REC-LEN PIC S9(4) COMP.
    000101 03 VAR-LENGTH-DATA PIC X
    000102 OCCURS 1 TO 100 TIMES
    000103 DEPENDING ON VAR-REC-LEN.
    000104 COPY DFHAID.
```

The definition of VARIABLE-LENGTH-RECORD shows two fields: VAR-REC-LEN and VAR-LENGTH-DATA. Note that the length of VAR-LENGTH-DATA depends on the value of VAR-REC-LEN.

Now we will look at the current contents of VARIABLE-LENGTH-RECORD. You can do this directly from the Working-Storage listing.

4. Tab to the line containing VARIABLE-LENGTH-RECORD.
5. Type d (for display), and press Enter.

```
CA InterTest for CICS V9.1 - PROSYM FILE UNCOND BEFORE BREAKPOINT
COMMAND ===> Program= COBDEMO Option # Stmt # Margin= 01
Search= +00100.

____ VAR-REC-LEN | +00100.
  000097 OCCURS 5 TIMES
  000098 INDEXED BY DISTRICT-X.
d 000099 01 VARIABLE-LENGTH-RECORD.
  000100 03 VAR-REC-LEN PIC S9(4) COMP.
  000101 03 VAR-LENGTH-DATA PIC X
  000102 OCCURS 1 TO 100 TIMES
  000103 DEPENDING ON VAR-REC-LEN.
  000104 COPY DFHAID.
```
CA InterTest for CICS displays the contents of VARIABLE-LENGTH-RECORD in a structured format.

As you can see, the structured format gives only one occurrence of each field; it does not tell you the record length. To see the length of a 01-level record, switch from the structured format to the dump format.

6. Press Enter.

CA InterTest for CICS displays the contents of VARIABLE-LENGTH-RECORD in a hexadecimal dump format.

The dump format shows all occurrences of lower level fields and thus the 01-level record length. This display shows that the record length is 64 hexadecimal (or 100 decimal) bytes. The first two bytes are the field VAR-REC-LEN, and the rest of the bytes are repeated occurrences of the field VAR-LENGTH-DATA.

Now return to the breakpoint to continue execution
7. Press Clear to return to Source Listing Breakpoint screen. 
   Now you will execute one verb and then stop. This lets you see the contents of the 
   record after the demo program changes its length.

   The demo program executes one verb (the MOVE statement, which moves 48 to 
   VAR-REC-LEN). Then CA InterTest for CICS immediately stops the program and 
   displays another breakpoint.

   Now the program is halted at the next statement, as shown next.

   ![CA InterTest for CICS V9.1 - PROSYM FILE STEP BEFORE BREAKPOINT](image)
   
   Program= COBDEMO  Option #       Stmt #  Margin= 01
   Search=
   
   000688**** CAN LOOK AT VARIABLE LENGTH DATA **********
   000689     MOVE +48 TO VAR-REC-LEN.
   000691     ==>     GO TO SEND- MAP00.
   000692 REP FILE.
   000693     IF TASK-SWITCH3 EQUAL SPACE.

9. To display the contents of VARIABLE-LENGTH-RECORD from the Source Listing 
   screen, you would have to find a line containing the field (such as the definition), 
   type d, and press Enter. This is what you did the first time. Another method is to use 
   a CORE menu:

   CA InterTest for CICS displays the Breakpoint Primary Option Menu.

11. Select option 1  Main Menu.
    CA InterTest for CICS displays the Primary Option Menu.

12. From the Primary Option Menu, select option 3 Main Storage.
    CA InterTest for CICS displays the Main Storage Menu.
13. Select option 3 Breakpoint areas.

CA InterTest for CICS displays a CORE menu for breakpoint-related data.

```
CA InterTest CORE COMMAND BUILDER - BREAKPOINT-RELATED AREAS (CORE=Bkpt)
Specify area to be displayed, changed or moved: Task number: 00094

COBOL name: variable-length-record

Special element: (Enter highlighted keyword)
  SCCR Saved screen image  BMSG Breakpoint message  CMAR Commarea
  FCAR Facility cntl area  EIB Exec interface block  TURAR TCT user area
  BLLS COBOL BLL cells  OM: COBOL working storage  TGT COBOL TGT
  LCL COBOL local storage  DSA COBOL DSA

SCAN VALUE: ________________________________ DATA FORMATS
SCAN RANGE: ___ B to scan backwards: __
To VERIFY and/or CHANGE Data:
Existing data: ________________________________
New data: ________________________________

MOVE From ________________________________ To ________________________________

PF1 Help  2  3 End  4 Return  5  6
PF7  8  9 Complex  10  11  12
```

14. Type `variable-length-record` in the COBOL name field, and press Enter.

CA InterTest for CICS displays the modified contents of `VARIABLE-LENGTH-RECORD` in structured format.

```
CA InterTest for CICS V9.1 - MAIN STORAGE UTILITY - Termid = U031
Starting at Address =2080A6F0 Structure Display Format
01 VARIABLE-LENGTH-RECORD |
02 VAR-REC-LEN | +00048. |
02 VAR-LENGTH-DATA |

```

15. Press Enter to display the record in dump format.

The record is now only 30 hexadecimal (or 48 decimal) bytes long because the variable field was reduced from a length of decimal 100 to decimal 48 when the program moved 48 to VAR-REC-LEN.
   CA InterTest for CICS displays the Main Storage Menu.
17. Press PF4 until the Source Listing Breakpoint screen displays.
18. Press PF5.
   The demo program resumes execution and displays the Options Menu.

Review What Happened

In this part of the demo session you displayed the length and contents of a variable length record.

- You used an unconditional breakpoint to halt program execution before the demo program changed the length of a variable length record.
- You inspected the contents of the record.
- You then executed one verb, halted program execution, and again inspected the contents of the record. This enabled you to confirm its new length.

The ability to display a record before and after a program instruction executes makes it easy to check program logic.

Note: You have now completed Option 04 of the advanced demo session.
1. Press Enter to continue with the demo session.
   or
   Press Clear to terminate the session.

Option 05: How to Work with Indexed Table Items

This section of the demo session shows you a fast and easy way to inspect and modify the contents of indexed data items.

Note: Before you begin this section, complete the steps outlined in Demo Preliminaries (unless you did so earlier in this session). The CA InterTest Demo Session Options Menu should be displayed.
1. Select option 05, and press Enter.

   The demo program displays a screen that describes what occurs in this part of the demo session.

   **********************************************************************
   ****                  CA InterTest Demo Session                   ****
   ****                    Work with Table Items                     ****
   ****                  CA InterTest Demo Session                   ****
   **********************************************************************

   CA InterTest makes it easy to display values for indexed, subscripted and qualified data areas. A table defined in WORKING-STORAGE has not been fully initialized. Here is what will happen:

   1. CA InterTest halts the program at an automatic breakpoint when it detects an uninitialized table item.
   2. You display the table item and correct the contents on the CORE display. You also display the contents of the indexes to determine which table entry caused the error.
   3. COBDEMO successfully completes execution.

   When this section of the demo continues, you will return to the Options Menu.

   Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

   The demo program resumes execution.
The next screen displayed is an automatic breakpoint screen. CA InterTest for CICS halted the demo program because it detected a data exception in the highlighted ADD instruction.

If you recall, this is the same type of error detected and corrected in the basic demo session. To continue execution, you need to find the data item causing the ASRA and dynamically modify it.

Looking at the highlighted breakpoint statement, you might suspect the variable is at fault; it probably was not initialized correctly. The variable in this case is an indexed table entry.

**Display an Indexed Table Entry**

You can quickly inspect and correct the value of the field DISTRICT-POP (STATE-X, COUNTY-X, DISTRICT-X) directly from this screen in the AutoKeep display at the top or by entering `d` and positioning the cursor under the field name within the code.

1. Overtype the A to the left of the highlighted statement with `d` (for display).
2. Place the cursor under any letter in DISTRICT-POP, and press Enter.

CA InterTest for CICS displays the current value of DISTRICT-POP in a structured format.

The error message on the bottom line in the following screen tells us this field does not contain a valid value, according to its definition. This is the error that triggered the automatic breakpoint.
Correct an Uninitialized Table Item Dynamically

To dynamically correct this error, modify main storage so the field contains a packed decimal zero. This is the hexadecimal value: 000C. Do this by overtyping the value displayed.

1. Tab once to go to the hexadecimal display area.
2. Move the cursor to the final 0, overtype it with c, and press Enter.

CA InterTest for CICS dynamically modifies the value in main storage for DISTRICT-Pop (STATE-X, COUNTY-X, DISTRICT-X), and displays the new value.

The previous error message no longer displays. You could now resume execution from the breakpoint because you dynamically corrected the error.

Before you do, first look at the CORE command displayed above the message line:

```
CORE='DISTRICT-Pop(STATE-X, COUNTY-X, DISTRICT-X)'
```

CA InterTest for CICS formatted this command when you used the d and the cursor to display the table item. Using the d and the cursor meant you did not have to type the complex name. Whenever you can, let CA InterTest for CICS do the work for you.

Display Values of Indexes

Although you corrected the error, you do not know exactly which table entry was at fault. To learn this you will need to know the current contents of STATE-X, COUNTY-X and DISTRICT-X and their occurrences in the table.

1. Return to the breakpoint display to request a display of STATE-X.
2. Press PF3.

CA InterTest for CICS displays the previous Source Listing Breakpoint screen.

The program code directly above the breakpoint line shows the values of COUNTY-X and DISTRICT-X. You also could page backward to see the value of STATE-X. However, suppose this code was not there. How would you find the values of STATE-X, COUNTY-X and DISTRICT-X? First, we will see how to display the contents of the variable STATE-X.

3. Tab to the AutoKeep line containing STATE-X.

4. Type d.

5. Press Enter.

CA InterTest for CICS displays the current contents of STATE-X in dump format, as illustrated in the following figure. CA InterTest for CICS displays the values of indexes in dump format rather than structured format because indexes are not defined in a structure.

STATE-X has a value of hexadecimal zero.

The command CA InterTest for CICS generated is simply CORE='STATE-X', as shown on the bottom of the previous figure. This confirms the dump display is showing you the main storage contents of STATE-X, which is what you wanted.

The last line on the screen indicates STATE-X is set to occurrence 1.
To see the contents of the remaining index variables, COUNTY-X and DISTRICT-X, you could return to the breakpoint display and repeat the request using the d and the cursor method. However, a quicker method is to modify the CORE command on the bottom of the screen. All you need to do is change the STATE-X in the CORE command to COUNTY-X.

6. Move the cursor to the CORE command.

7. Overtype 'STATE-X' with 'COUNTY-X', and press Enter.

CA InterTest for CICS displays the contents of COUNTY-X in dump format and informs you that it is set to occurrence 3, as shown in the following screen. (Note the modified command.)

```
6. Move the cursor to the CORE command.
7. Overtype 'STATE-X' with 'COUNTY-X', and press Enter.
CA InterTest for CICS displays the contents of COUNTY-X in dump format and informs you that it is set to occurrence 3, as shown in the following screen. (Note the modified command.)

8. To see the value of DISTRICT-X, modify the CORE command to specify DISTRICT-X instead of COUNTY-X.
9. Move the cursor to the CORE command.
10. Overtype 'COUNTY-X' with 'DISTRICT-X', and press Enter.
CA InterTest for CICS displays the contents of DISTRICT-X and informs you that it is set to occurrence 5. (Note the modified command.)

By displaying the contents of STATE-X, COUNTY-X, and DISTRICT-X, you have determined that DISTRICT-POP (1, 3, 5) is the exact position in the table of the invalid table item.

Now return to the breakpoint display.
   The Source Listing Breakpoint screen redisplay.

   The demo program resumes execution and returns you to the Options Menu.

Review What Happened

In this part of the demo session the following occurred:

- The demo program was halted at an automatic breakpoint because a table item contained invalid data.
- You corrected the invalid data. However, you did not know the exact position in the table of the invalid table item.
- You used CORE to display the contents of the indexes and their occurrences. This enabled you to pinpoint the position of the invalid table item.

With CA InterTest for CICS, it is easy to debug problems involving indexed, qualified, or subscripted data items because you can display and modify their contents just as you would any other COBOL data item.

You have now completed Option 05 of the advanced demo session.

1. Press Enter to continue with the demo session.
   or
   Press Clear to terminate the session.
Option 06: How to Detect a Storage Violation

This section of the test session details how CA InterTest for CICS detects and prevents a storage violation and lets you continue testing.

Note: To perform this section, you should have completed the steps outlined in Demo Preliminaries. The CA InterTest Demo Session Options Menu should be displayed.

1. Select option 06, and press Enter.

   The demo program displays a screen that describes what occurs in this part of the test session.

   The program attempts to move data from one field to another. However, because the program has relinquished storage for the receiving field, a storage violation would occur. Instead, here is what will happen:

   1. CA InterTest halts the program at an automatic breakpoint.
   2. You inspect the FREEMAIN instruction which released the storage for the field.
   3. You return to the breakpoint display. Then you go around the problem, continuing program execution from the statement following the one that triggered the breakpoint.

   When this section of the demo completes, you will return to the Options Menu

   Press ENTER to continue or CLEAR to terminate.

2. Press Enter.

   CA InterTest for CICS halts the demo program at an automatic breakpoint.
Prevent a StorageViolation

Displayed next is a breakpoint screen. CA InterTest for CICS halted the demo program at an automatic breakpoint. CA InterTest for CICS prevented the storage violation that would have resulted if the highlighted MOVE statement had been executed. CA InterTest for CICS also inserted an A next to the statement to indicate an automatic breakpoint.

| CA InterTest for CICS V9.1 - PROSYM FILE ABEND DETECTED BREAKPOINT |
|------------------|------------------|------------------|
| Command | COBDEMO | Option # | Stmt # | Margin= 01 |
|---------------------------------------------|
| Search= | | | | |
| NEW-DATA | | | | |
| STG-AREA1 | | | | |
| 000968**** NOW MOVE NEW DATA INTO ACQUIRED AREA |
| 000969* |
| A | MOVE NEW-DATA TO STG-AREA1. |
| ===> | | | | |
| ===> | an attempt to change an area that does not belong to this task. |
| ===> | Possible system damage has been prevented. |
| ===> | |
| ===> | Press PF1 for a detailed description. |
| 000971 | GO TO SEND-MAP00. |
| 000972 | WHICH-ONE. |

The ability of CA InterTest for CICS to prevent storage violations is one of the most important reasons to monitor programs. Storage violations are notorious CICS errors that can do the following:

1. Bring down your CICS system.
2. Cause a program to produce erroneous data.
3. Cause one program to corrupt the data of another program. In this case, tracking down the problem without CA InterTest for CICS is almost impossible because the error is not in the affected program.

By stopping the program before the storage violation can occur, CA InterTest for CICS protects your system and helps you diagnose and correct the error.

In this example, the statement that would have caused the storage violation is: MOVE NEW-DATA TO STG-AREA1. This statement moves data from NEW-DATA to STG-AREA1. Perhaps the program does not own the area of storage defined as STG-AREA1, so the program cannot move data to that field.
Inspect the FREEMAN Statement

We will page back to find out what caused the error.

1. Press PF7.

CA InterTest for CICS displays the following screen.

Look at the FREEMAIN statement that releases the storage for STG-AREA. Now it is clear why the storage violation occurred. The demo program released the storage for STG-AREA. After releasing the storage, the program tried to move data to STG-AREA1, which is part of STG-AREA.

To correct the problem, you would have to remove the FREEMAIN statement or change its location, and then you would recompile the program. However, one of the benefits of CA InterTest for CICS is that you do not have to correct a problem when you discover it. Instead, you can go around the problem and continue testing the program.

Now we will redisplay the breakpoint.

2. Press Clear.

CA InterTest for CICS redisplay the automatic breakpoint.
Resume Program Execution

We will continue program execution at the first instruction after the statement that triggered the breakpoint. In the COBOL demo, this is GO TO SEND-MAP00. In the COBOL/II and COBOL/370 demos, this is a GOBACK statement.

1. Tab to the first statement following the breakpoint.
2. Enter a g (go).
3. Press Enter to continue program execution.

CA InterTest for CICS resumes program execution. The demo program successfully completes execution and displays the Options Menu. By going around the storage violation, you have allowed the demo program to continue executing despite the problem CA InterTest for CICS detected.

Review What Happened

In this part of the demo session the following occurred:

- CA InterTest for CICS detected and prevented a storage violation and halted the demo program at an automatic breakpoint.
- The ability of CA InterTest for CICS to prevent storage violations protects your CICS system and prevents one program from corrupting another.
- By inspecting the statement that would have caused the storage violation, you saw which field was in error. By inspecting the FREEMAIN statement, you were able to determine why the program did not own the storage area.
Option 07: How to Test a Composite Module

Using CA InterTest for CICS to examine the source code surrounding an error makes it easy to learn what caused the problem.

You went around the problem and continued program execution.

The ability to go around a problem lets you continue testing and correct many errors in a single test session.

Note: You have now completed Option 06 of the advanced demo session.

1. Press Enter to continue with the demo session.
   or
   Press Clear to terminate the session.

Option 07: How to Test a Composite Module

This option details how you can use CA InterTest for CICS to test and debug a composite module. Composite support is an CA InterTest for CICS feature designed to help programmers who are responsible for testing and debugging called subroutines. Skip this option if you are not responsible for these subroutines.

To perform this option, you should have set monitoring for the composite module as described in Composite Support (see page 94) part of Demo Preliminaries.

In this section you will test three programs:

- The main program, COBDEML, the composite module is invoked by the same DEMC transaction as the demo program COBDEMO.

- Two subroutines ASBIN25 and CSBIN25 that are part of composite module COBDEML. However, composite modules at your site could have many called subroutines. You need to tell CA InterTest for CICS only about the ones that you want to test or debug.
Follow these steps:

1. In the CA InterTest Demo Session Options Menu, select option 07, and press Enter.
   The following screen appears:

   Program COBDEML calls two subroutines: ASBIN25 and CSBIN25.
   1. CA InterTest halts the program at an automatic breakpoint caused by an ASRA. Without composite support, you will see only that you are stopped at the CALL to ASBIN25.
   2. You abend the task, and then set composite support for COBDEML.
   3. Type in transaction id DEMC again. This time you will see the actual instruction causing the ASRA within the subroutine.
   4. You correct the problem that caused the automatic breakpoint.
   5. You then set an unconditional breakpoint in CSBIN25. When you resume execution, the program is halted at that breakpoint.
   6. You remove the breakpoint and the program completes execution.

   When this section of the demo completes, you will return to the Options Menu. Press ENTER to continue or CLEAR to terminate.

2. Press Enter.
   The composite module halts at an automatic breakpoint at statement CALL ASBIN25.

   Note: If a screen other than the following screen displays, someone at your site has already set composite support for the composite module. In this case, skip to Re-execute the Demo Program (see page 141).
CA InterTest for CICS halted the composite module at an automatic breakpoint because it detected and prevented an error in ASBIN25 when control passed to that subroutine. However, so far you do not know where in ASBIN25 the error occurred.

**Note:** The name in the Program field is the name of the composite module COBDEML. This is the program CA InterTest for CICS is monitoring—the main composite program for which you set monitoring before you began the advanced demo session.

Setting composite support lets you test and debug a called subroutine as if it were a separate program, with full symbolic support. That means you can set breakpoints and other monitoring options for that subroutine. When CA InterTest for CICS detects an error, it generates an automatic breakpoint at the statement that triggered the abend, not at the `CALL`.

You can set composite support before you begin to test a program or at any time during testing. In this case, you are going to abend the task, set composite support, and then resume testing.

**Abend the Task**

When your program is stopped at a breakpoint, you can abend the task rather than resume execution.

1. Type abend in the Command line of the Breakpoint display, and press Enter.

   **Note:** This is the same as selecting option 3 Abend, from the Breakpoint Menu.

   CA InterTest for CICS displays the screen shown next.

   ```plaintext
   ------------------
   CA InterTest v9.1 ABEND BREAKPOINTED TASK
   ------------------
   COMMAND =>
   
   Type an abend code, then press ENTER.
   
   Abend Code ___  Abend code options are:
   
   blanks  Normal abend, no dump
   XXXX    Abend exits cancelled, no dump
   your code  Your abend code, dump taken
   .
   .
   .
   `
   ```

   Because you do not need a dump, you do not need to enter an abend code.

2. Press Enter.

   CICS informs you the task has been abended.

### Setting Composite Support

To set composite support, access the main CNTL menu from which you can set all of CA InterTest for CICS monitoring options.

1. Type `itst` on a clear screen, and press Enter.
   CA InterTest for CICS displays the Primary Option Menu.

2. Select option 2 Monitoring.
   CA InterTest for CICS displays the Monitoring Menu.

3. Select option 1 Programs.
   CA InterTest for CICS displays the Program Monitoring Menu.

```
--------------- CA InterTest V9.1 PROGRAM MONITORING ---------------
COMMAND ===>

Type information and S to set or R to remove option(s) below.
Program . . cobdeml_ Program name (or .ALL, .OPTIONS or generic)
User ID . . _______ User (or .ANY) for whom the program is monitored
Option       Description                                      More:   +
_ Status       Display and/or remove monitoring options (S only)
_ Monitor      Monitoring (R removes monitoring and all options previously set)
_ UBP          Unconditional breakpoints (specific program only)
_ CBP          Conditional breakpoints (specific program only)
_ RBP          Breakpoints for CICS, DB2, DL/I or external CALL requests
_ Stmt Trace   Statement tracing and data monitoring (COBOL only)
_ New copy     Fetch new copy of program and reset monitoring options (S only)
_ Commands     Indirect commands defined for a specific COBOL or PL/1 program
_ Replace      CICS resource name replacement options
_ Protect      Storage protection monitoring options
_ Special      Other options (storage allocation, file updating, etc.)
_ Composite    Monitor multi-CSECT program's separately compiled components
PF1 Help       2            3 End        4 Return     5          6
PF7 Backward   8 Forward  9           10           11         12
```

4. In the Program field, type the name of the composite module you are using `COBDEML`.

---
5. Type $ to the left of Composite.

CA InterTest for CICS displays the Composite Support menu.

```
CA InterTest for CICS V9.1 - Composite Support Builder
COMMAND ===> SCROLL: PAGE
Define composite support for COBDEML and press PF5. Row 00001 of 00003

* * * * * * * * * * *
<table>
<thead>
<tr>
<th>Link name</th>
<th>Monitor</th>
<th>Offset</th>
<th>Length</th>
<th>Language</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>S COBDEML COBDEML</td>
<td>20</td>
<td>748</td>
<td>IBMCOB 4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S ASBIN25 ASBIN25</td>
<td>768</td>
<td>17C</td>
<td>HLASM 6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S CSBIN25 CSBIN25</td>
<td>8E8</td>
<td>674</td>
<td>IBMCOB 4.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

This menu displays link-edit information for the composite program and its subroutines; for example, ASBIN25 and CSBIN25. This information is received from a batch job, executed after the composite module was link-edited.

The batch job that provides the link-edit information identifies all of a program's called subroutines. For efficiency, however, it is a good idea to specify monitor-names just for the subroutines you want to test. You can always set monitoring for additional subroutines by assigning monitor-names on the Composite Support menu.

6. Press PF5 to enable composite module monitoring.

CA InterTest for CICS sets composite support for COBDEML and returns you to the Program Monitoring menu.

Now CA InterTest for CICS monitors the subroutines and the composite module so you can take advantage of all CA InterTest for CICS facilities to test and debug each of the programs.

**Re-execute the Demo Program**

Now you can re-execute the demo program.

2. CA InterTest for CICS displays the Primary Option Menu.
3. Select option X Exit to exit to CICS.

4. Clear the screen and type the transaction ID DEMC for the COBOL demo program and press Enter.

CA InterTest for CICS displays the automatic breakpoint at which you were initially halted. However, now the breakpoint occurs at the actual statement in ASBIN25 that triggered the abend, not at the CALL to ASBIN25.

CA InterTest for CICS V9.1 - PROTSYM FILE SOURCE LISTING BREAKPOINT

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJECT CODE</th>
<th>ADDR1</th>
<th>ADDR2</th>
<th>STMT</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000BA</td>
<td>0000BA</td>
<td>FA20</td>
<td>D1A0</td>
<td>211B</td>
<td>TASKNUM,=P’1’</td>
</tr>
</tbody>
</table>

Note: The name in the Program field is ASBIN25. When control passes to ASBIN25, CA InterTest for CICS starts monitoring that program because you set composite support. Setting composite support enables CA InterTest to identify and display the statement in the subroutine where the error occurred.

The error occurred because of improperly formatted data in TASKNUM. The AP instruction tries to move a packed decimal 1 to TASKNUM. TASKNUM is defined as a packed decimal field, but it contains binary zeros instead of a valid packed decimal value. This is the same type of error you corrected in the basic demo session.

**Display the Data in TASKNUM**

To correct this error, display the contents of TASKNUM and dynamically change the value.

1. Tab to the AP instruction, and overtype the A with d.

2. Position the cursor under any character in TASKNUM, and press Enter.

   CA InterTest for CICS displays the contents of TASKNUM.

   On the left of the screen are the Assembler data field names; on the right are the corresponding hexadecimal and character representations of each field.
CA InterTest for CICS displays more than just the contents of TASKNUM. It displays TASKNUM and all the fields below it in the same data structure; that is, in the same DSECT.

Correct the Data in TASKNUM

As you can see, TASKNUM does not contain a packed decimal value. TASKNUM in the COBOL version contains binary zeros rather than a packed decimal value; for COBOL II and COBOL/370 programs, TASKNUM contains other, incorrect values.

You can correct the error by overtyping the incorrect value with a correct value.
1. Tab to the top line and move the cursor to the Hexadecimal value of TASKNUM.
2. Overtype the current value; set it to a packed decimal zero by tying zero over the "?" and erase the remaining data (erase EOF).
3. Press Enter.

CA InterTest for CICS redisplays the screen, changing your entries to uppercase.

Now TASKNUM contains a packed decimal zero. You have corrected the error that triggered the automatic breakpoint.


CA InterTest for CICS redisplays the previous Source Listing Breakpoint screen. However, the explanation of the abend no longer displays.
Set a Breakpoint in Subroutine CSBIN25

Now you are going to set a breakpoint at the first statement in subroutine CSBIN25, the other subroutine you asked CA InterTest for CICS to monitor. Then, when you resume execution, CA InterTest for CICS halts execution at the breakpoint in this subroutine.

When you are stopped in a subroutine, you can use all of the CA InterTest for CICS features, including the following:

- View the program listing and compiler output
- Display and modify main and auxiliary storage
- Set and remove breakpoints and other monitoring options
- Request single-step execution
- Display the backtrace that brought the program to its current point

1. Tab back to the Program field.
2. Overtype the current name with the COBOL subroutine name CSBIN25 and press Enter.

CA InterTest for CICS displays the source listing for COBOL program, for example, CSBIN25.

Now you are going to set an unconditional breakpoint at the first statement in the Procedure Division.

3. Tab to the MOVE ZERO TO DIVCT statement.
4. Type a u to the left of that statement, and press Enter.

```cobol
CA InterTest for CICS V9.1 - PROSYM FILE ABEND DETECTED DISPLAY
COMMAND ===> Program= CSBIN25 Option # Stmt # Margin= 01
Search=

000017     05 COMM-DATE.
000018     10 COMM-MM PIC 99.
000019     10 FILLER PIC X.
000020     10 COMM-DD PIC 99.
000021     10 FILLER PIC X.
000022     10 COMM-YY PIC 99.
000023 PROCEDURE DIVISION USING COMM-TEXT.
          u 000024     MOVE ZERO TO DIVCT.
          000025 GET-DATE.
          000026     MOVE COMM-YY TO PROGYY.
          000027     MOVE COMM-MM TO PROGMM.
          000028     MOVE COMM-DD TO PROGDD.
          000029     DIVIDE PROGYY BY 4 GIVING DIVYY REMAINDER YY-REM.
          000030     MOVE PROGMM TO DIVMM.
          000031     GOBACK.
An "M" preceding a data-name reference indicates that the data-name is modified

  DEF DATA NAMES REFERENCES
    17 COMM-DATE
```

CA InterTest for CICS sets an unconditional breakpoint at the MOVE statement.

5. Press Clear.

CA InterTest for CICS redisplays the previous Source Listing Breakpoint screen.

**Resume Program Execution**

Now you are going to continue program execution.

1. Press PF5.

The program resumes execution. When the control passes to subroutine CSBIN25, CA InterTest for CICS halts the program at the unconditional breakpoint you just set.

At this point you could use CA InterTest for CICS to examine the program listing of CSBIN25 or the values of any program variables. However, you are simply going to remove the breakpoint and resume program execution.

**Remove the Breakpoint and Resume Execution**

1. Overtype the U with x, and press PF5.

CA InterTest for CICS removes the unconditional breakpoint and program execution resumes. COBDEML completes execution and the Options Menu is displayed.
Review What Happened

In this part of the demo session the following occurred:

- When you executed COBDEML, CA InterTest for CICS detected and prevented an abend caused by invalid data in one of its called subroutines.

  CA InterTest for CICS displayed the automatic breakpoint at the statement that called the subroutine, rather than the statement that triggered the breakpoint, because CA InterTest for CICS was not monitoring the subroutine.

- You set composite support for COBDEML so that CA InterTest for CICS would monitor its subroutines: ASBIN25 and CSBIN25.

  Setting composite support lets you test and debug a called subroutine with full symbolic support and use all CA InterTest for CICS features.

- When you re-executed COBDEML, CA InterTest for CICS redisplayed the automatic breakpoint; this time at the actual statement in subroutine ASBIN25 that caused the error.

- You corrected the error by overtyping the contents of the invalid field.

- You then set an unconditional breakpoint in the other subroutine, CSBIN25, and resumed program execution. When control passed to CSBIN25, CA InterTest for CICS halted execution at the breakpoint you set.

  The ability to halt execution at any statement in a subroutine means you can test a subroutine just as if it were a main program.

- You removed the unconditional breakpoint and continued program execution.

- You have now completed Option 07 of the advanced demo session.

1. Press Enter to continue with the test session.
   or
   Press Clear to terminate the session.
Index

A
abending tasks • 59
automatic breakpoints
  AEIL • 100
  CICS requests, excessive • 105
  Help • 61
  storage violation • 134
auxiliary storage
  displaying • 54
  inspecting • 53, 115
  modifying • 55, 117
B
backtrace
  automatic stepping • 74
  bookmark • 72
  ending a session • 75
  examining • 71
  navigating through execution • 74
  reading • 72
  See also • 71
    examiningSource Listing Backtrace • 71
Backtrace Summary screen • 71
bookmark, assigning • 72
breakpoints
  activities • 39
  overview • 39
  removing, status display • 31
  setting • 27
  single-stepping • 58
  types • 39
  variable-change • 50
BTRACE command • 72
breakpoints in subroutines, setting • 144
description • 86
setting • 137
testing • 137
conditional breakpoints
  removing • 67
  setting • 65
  variable-change breakpoints • 39
cross reference • 42
CSA, modifying • 85
CWA, modifying • 85
D
data items
  adding to Keep window • 48
definition, displaying • 42
displaying • 45
displaying in Keep window • 24, 47
displaying structure from Keep window • 49
modifying • 25, 46
modifying in Keep window • 49
MOVE statement, using • 46
removing from Keep window • 50
unconditional breakpoints, setting • 42
variable-change breakpoint, setting • 50
databases
  inspecting • 53
  modifying • 53
debugging tasks • 35
demo session, advanced
  CICS requests, limiting • 103
  CICS storage, limiting • 103
  COBDEMO, executing • 97
  composite support • 137
  FILE • 115
  File Control ID, replacing • 99
  indexed table items, displaying • 125
  No File Updating • 110
  overview • 89
  preliminaries • 90
  request breakpoints • 110
  request breakpoints, setting • 92
  storage violation, preventing • 133
  variable length data, displaying • 119
demo session, basic
automatic breakpoint • 23
data item, displaying • 24
data item, modifying • 25
Keep window • 24
monitoring, setting • 19
objectives • 13
program execution • 22
resuming execution • 28
source selection • 15
status display, removing breakpoints from • 31
test scenario • 13
unconditional breakpoints, setting • 28
dumps
managing • 87
symbolic • 87

E
ending a test session • 61
execution
path • 71
resuming • 28
exit, ITST menus • 20

F
FILE
capabilities • 117
facility • 115
files
displaying • 115
modifying • 53
not updating • 85, 110
replacing • 85, 99
FOL option • 85

H
Help
CA InterTest for CICS • 59
correcting errors • 61

I
indexed table items
displaying • 125
modifying • 129
indirect commands
attaching to breakpoints • 82
coding statements • 79
using • 79
ITST Primary Option Menu
demo session, basic • 15
exiting • 20
PF keys • 15

K
Keep window
data items, adding • 24, 48
data items, displaying structure • 49
data items, keeping • 47
data items, modifying • 49
data items, removing • 50

L
load modules, modifying • 85

M
main storage
inspecting • 45
limiting • 85
modifying • 46
modifying, overriding rules for • 85
protecting • 85
Monitor, source listing command • 19
Monitoring options
protection • 85
replacement • 85
special • 85
types • 84
Monitoring Status display, removing breakpoints from • 31
monitoring, setting • 19, 38
MOVE statement • 46
MXR option • 103
MXS option • 85, 103

N
navigate the Statement Trace table • 77
New Program Copy • 62
No File Updating • 85, 110

O
Option # field
displaying sections of program • 37
searching forward and backward • 37
P

Primary Option Menu See ITST Primary Option Menu • 15
Procedure Names • 43
Profile, source listing command • 19
program execution, navigating through • 74
programs
  controlling execution • 27
  executing • 38
  monitoring • 38
  new copy • 62
  replacing • 85
  resuming execution • 57
  testing, basics • 38
protection options • 85

R

Replacement options • 85, 99
request breakpoints
  removing • 68
  screen • 110
  setting • 68, 92
resuming execution • 28, 57

S

single-stepping • 58
Source listing
  Cross Reference, displaying • 42
displaying • 36
paragraph names, displaying • 43
profile options • 19
selection list • 15
source selection • 15
Source Listing Backtrace
  reading • 74
  viewing • 73
Source Listing Profile • 19
Source Listing Selection screen • 15
Source Menu • 15
special options • 85
Statement Trace facility
  navigate the Statement Trace table • 77
  overview • 76
  statement tracing and data monitoring • 76
  view past data values • 78
statement tracing and data monitoring • 76
status, source listing command • 20
storage violation, preventing • 133
subroutines
  testing • 86
  unconditional breakpoints, setting • 144
synchronized processing
  maintaining • 11
  Symbolic Version List • 11
  warning message • 11
T
tasks, abending • 59
temporary storage
  inspecting and modifying • 53
replacing • 85
transient data
  inspecting and modifying • 53
replacing • 85

U

unconditional breakpoints
  removing • 40
setting • 28, 90
setting at data name references • 42
setting at paragraph names • 43
setting in subroutines • 144
setting, at all labels and all procedures • 43
setting, common locations • 40
setting, on source listing • 40

V

variable length data, displaying • 119
variable-change breakpoints
  definition • 27
removing • 53
setting • 50

W

wild branches • 85