CA InterTest™ for CICS

User Guide

r8.5

Second Edition

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2012 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA products:

- CA InterTest™ for CICS
- CA Optimizer®
- CA Optimizer®/II
- CA SymDump® for CICS

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- File Functions for VSAM Records (see page 469)—Updated the information in this section to include XRBA.
- Removed the topic STR: Save the CICS Trace Table.
- Removed all references to FCT and FILE commands.
- Display Storage Related to a Task at Breakpoint (see page 420)—Modified the description for CORE=BLLnn in the topic, and added new entries, CORE=BLXnn and CORE=BLXS to the table.
- The Detailed Breakpoint Screen (see page 178)—Added two subtopics in this section.
- Indirect Commands (see page 187)—Added information related to the Indirect Command RUN to the following topics: How Can You Use Indirect Commands (see page 187), Indirect Command Examples (see page 193), and Indirect Command Syntax (see page 199).
- Accessing Main Storage: CORE (see page 375)—Added a new topic, Display Storage for an Inactive COBOL Program at Breakpoint under Breakpoint Commands.
- Special Circumstances (see page 574)— Updated the information in this section.

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 23

What Is CA InterTest for CICS? 23
Fast, Easy Testing 24
Interactive Testing: The CA InterTest for CICS Monitor 24
Error Detection and Prevention 25
Diagnostic Information....26
Interrupt Programs by Setting Breakpoints 27
Stable CICS Test and Production Systems 28
Examine and Modify Main and Auxiliary Storage 29
Easy-to-Use Features....29
Getting Started with CA InterTest for CICS 30
Different Methods of Using CA InterTest for CICS 30
Get HELP....31
User-Defined Help for Abends 32
Online Access to Source Listings and Compiler Output 32
Symbolic Support 32
Flexible Testing....32
CA InterTest for CICS Components....33
CA SymDump for CICS 33
Take Advantage of CA InterTest for CICS 34

Chapter 2: Menus and Displays 35

The Primary Option Menu and Breakpoint Menu....36
Access....36
Whats New?....37
Breakpoint Primary Option Menu....37
Access....37
Option Selection....37
P F Keys 38
Menu Structure and Navigation....39
Breakpoint Menus....40
Selection and Navigation Commands 41
The Monitoring Status Display 42
Sample Status....43
Access....44
Expand and Collapse Entries 45

Remove Monitoring, Breakpoints, or Other Options 45
Refresh the Display 46
Exit the Status Display 46

Chapter 3: Source Listing Facility 47

Prepare Your Program 47
Start CA InterTest in Your CICS Region 47
CA InterTest for CICS Transaction IDs 48
Start a Test Session 49
Step 1. Perform CICS Sign On 49
Step 2. Display Your Source Code 50
Step 3. Set Monitoring 50
Step 4. Set Breakpoints 51
Step 5. Initiate Your Program and Begin Testing 51
The Source Selection List 51
Examine the Source Listing Display Screen 54
Detailed Field Definitions 54
Source Listing Option # Field Entries 56
Source Listing Command Line 57
Action-Characters Supported from a Source Listing Display or Breakpoint 63
Source Listing PF Keys 66
Search for Data 66
Find a Definition or Search Using the Search = Field 67
Search the Program Online Listing 68
Locate an Area Within the Program Online Source Listing 68
View the Online Listing 70
Jump to the Indirect Commands Build Screen 72
Jump to the Breakpoint Options Build Screen 72
Scroll Forward and Backward 73
Change the Program 73
Position the Display 73
Adjust the Margins 75
Customize the Source Listing Profile 76
Display Titles, No Titles, or Registers 77
Scroll Amount 80
Step Timing 80
Step Amount 81
Turn on Automatic Single-Stepping 82
Breakpoint Display Mode 83
From and BKPT Terminal IDs 84
Set the User ID 86

Set the Keep Window to Display Variables in Current Statement 87
Set Code Coverage 88
Set Structure Display Form 89
Testing Activities 89
Set Monitoring 89
Monitoring Multiple Programs 90
Display and Search Through Nested Programs 91
Maintain Synchronized Processing 92
COBOL and PL/I Symbolic Version Processing 93
Assembler Symbolic Version Processing 94
The Symbolic Version List 94
Composite Support for Duplicate Subprograms 98
Set Breakpoints 99
Remove Breakpoints 99
Statement Tracing 100
Data Monitoring 100
Run Your Program 101
Breakpoint Activities 101
Exit the Source Listing Facility 101

Chapter 4: Using Breakpoints 103

Set Breakpoints 103
Where Should You Set Breakpoints? 104
When Should You Set Breakpoints? 104
Why Should You Set Breakpoints? 104
Unconditional Breakpoints 105
Set Unconditional Breakpoints from the Source Listing 105
Set Unconditional Breakpoints from the Menus 108
Conditional Breakpoints 113
Set Conditional Breakpoints from the Source Listing 113
Set Conditional Breakpoints from the Menus 115
The Detailed Conditional Breakpoint Screen 121
Variable-Change Breakpoints 124
Request Breakpoints 124
Set Request Breakpoints from the Source Listing 125
Set Request Breakpoints from the Menus 125
Remove Breakpoints 129
Remove Breakpoints Flagged on Your Source Listing 130
Remove Unconditional Breakpoints from the Menus 131
Remove Conditional Breakpoints from the Menus 132
Remove Request Breakpoints 133

Chapter 5: Breakpoint Activities 135

Examine the Source Listing Breakpoint Screen 136
Source Listing Breakpoint Commands 137
Source Listing Breakpoint PF Keys 142
Respond to Automatic Breakpoints 143
Example 144
Bypass the Monitoring of a Statement that Triggered an Automatic Breakpoint 144
Temporary Override 145
Permanent Bypass 145
Work with Data at a Breakpoint 146
Search for Data 147
Find a Definition or Search Using the Search = Field 147
The Online Listing 148
Position Listing to a Hex-Offset 148
Position Listing to Current Breakpoint Processing 148
The Keep Window 148
Add a Storage Item 149
Using AUTOKEEP 150
Change an AutoKeep Item into a Permanent Keep Item 150
Scroll Forward and Backward 150
Scroll Left and Right 150
Remove Items 151
View and Change Main Storage Display from the Keep Window 151
Display or Modify Main Storage (CORE) 152
Display the Value of a Data Item 152
Use the Cursor 153
Example 154
Modify a Data Item by Overtyping the Main Storage Display 154
Example 156
Modify the Value of a Data Item with a MOVE Command 156
Example 157
The Statement Trace Facility 158
Enable Statement Tracing and Data Monitoring 158
Navigate the Statement Trace Table 158
View Past Data Values 160
Usage Notes 160
Display or Modify Auxiliary Storage (FILE) 161
FILE at a Breakpoint 162
FILE Changes Do Not Affect Main Storage at a Breakpoint 162
The Breakpoint Primary Option Menu 163
Access 163

PF1 Help 163
The CALLTRACE Facility 164
The CHANNEL Command 165
The Storage Facility 166
The Backtrace Facility 167
Access the Backtrace Facility 167
The Backtrace Summary 170
Example 170
Backtrace Summary Screen PF Key Assignments 171
The Source Listing Backtrace 171
Read the Source Listing Backtrace 172
Source Listing Backtrace PF Keys 173
Navigate Through Program Execution 174
Example 175
Example 176
Ending a Source Listing Backtrace Session 176
Additional Breakpoint Displays 176
The Detailed Breakpoint Screen 178
Read the Backtrace on the Detailed Breakpoint Screen 181
Continue Execution 182
Single-Stepping 182
Resume Execution 183
CNTL Commands 183
Continue Processing During Breakpoint Processing 184
Continue Processing During Breakpoint Processing Until Specified Verbs Are Executed 184
INTC--Continue Processing Until a COBOL PERFORM Has Been Completed Avoiding Breakpoints 185
INTC--Continue Processing After a COBOL Program Breakpoint Until the Next Call Or PERFORM Statement 185
Continue Processing During Breakpoint Processing Ignoring Preset Breakpoints 185
Continue Processing During Breakpoint One Verb at-a-Time 185
See More Code on the Detailed Breakpoint Display 186
Restore the Detailed Breakpoint Display 186
Switch Between Breakpoint Screens 186
Indirect Commands 187
How Can You Use Indirect Commands? 187
Overview of Steps 188
Set a Breakpoint to Execute an Indirect Command 189
Code Indirect Commands 192
From Source Listing 192
Indirect Command Examples 193
View Indirect Commands on the Status Report 196
Access 196

Edit Indirect Command Definitions 197
ITST Access to Indirect Commands 197
Indirect Command Syntax 199
Execution Flow Control 199
Command Syntax 200
Equate Command 200
Exit Command 200
Goto Command 201
IF, THEN Statement 201
Break and Pause Commands 202
Move Command 202
Perform Command 203
Format Indirect Commands 204
Data Names and Variables 205
For COBOL (prior to COBOL II Release 1.3) 205
For COBOL II (Releases 1.3 and higher) and COBOL/370 205
For PL/I Programs 206
Literals in Indirect Commands 207
Figurative Constants in Indirect Commands 207
Conditional Expressions in Indirect Commands 208
COBOL Variable Syntax 209
PL/I Variable Syntax 210
Examples 210

Chapter 6: Resume and Execution Options 213

Control and Resume Execution from a Breakpoint 213
Single-step Through Execution 213
Set Additional Breakpoints 213
Jump to Another Location 214
Continue Execution 214
Single-Step Through Execution 215
Jump to Another Location 215
Resume Program Execution Ignoring All Breakpoints 218
The Resume Menu 218
Resume at the Next Instruction 218
Resume Task from a Label 219
Resume Task from a Statement 219
Resume Task from an Offset 220
Resume Task from an Indirect Command Sequence 220
Resume Until Next EXEC/CALL 221
Resume Task for nnn Steps 221

Resume Task for nnn Machine Instructions 222
Abend the Task 222
Review Program Status After Execution 223
Your Program Source Code 223
CA InterTest for CICS Monitoring and Monitoring Options 224
Files and Databases 224
Continue to Test 225
Correct Your Source Code 225
Reset Breakpoints after Recompiling or Re-Assembling 226

Chapter 7: Monitoring Menu Options 227

Monitor Status 227
Breakpoints 227
Monitoring Menu Functions (ITST 2) 228
Programs 228
Access 229
PF Keys 230
The User ID Monitoring Option 230
User ID Set to .ANY 231
What Is Your User ID Default? 232
Override the User ID Default 233
Establish a Personal Debugging Session 233
Override the To and From Terminal Defaults for Breakpoints 233
Monitoring Submenus 234
Monitor Options for Programs, Transactions and Terminals 235
Initiate Monitoring 235
The Program Monitoring Submenu 237
Generic Specification of Programs, Transactions, or Terminals 238
Hierarchy Rules for Monitored Entries 239
Monitoring 240
Set Monitoring 240
Remove Monitoring 241
Set and Remove Statement Trace Options 242
Set Statement Trace Options 243
New Copy Option 244
Replacement Options 245
Set Replacement Options 245
Remove Replacement Options 246
Protection Options 248
Set Protection Options 249
Remove Protection Options 250

Special Options 252
Set Special Options 252
Remove Special Options 254
Composite Support 255
Set Composite Support Using the Batch Method 256
Change the Composite Support Screen 259
Set Composite Support Using the Online Method 260
Set Monitoring for a Program with Subprograms 261
Assembler Programs with Multiple CSECTs 262
Remove Composite Support 262
Disconnect, Reconnect, and Purge Active Tasks 263
Disconnect a Task 263
View and Set System-Wide Options 264

Chapter 8: Special Monitoring Situations 267

Special Monitoring Situations 267
Monitor Programs with LE/370 Condition Handlers 268
Monitor Dynamically Called COBOL II and COBOL/370 Programs 268
Production Environment Monitoring 269
Monitor Tasks That Execute Without Terminals 272
Monitor Tasks That Execute at Non-3270 Terminals 273
Segmented Monitoring to Handle Special Situations 274
Programmed Breakpoints Usage 287
Monitoring CICS/ESA 3.3 FEPI Applications 290
CA InterTest for CICS with IBM's EDF (EXEC Debugging Facility) 291
Advantages of CA InterTest for CICS Over EDF 292
Using CA InterTest for CICS with EDF 292

Chapter 9: CNTL Commands and Menus 295

CNTL Monitoring Commands and Options 295
CNTL Commands at a Glance 295
Hierarchy Rules for Monitored Entries 300
Initialize and Terminate CA InterTest for CICS 301
Restart CA InterTest for CICS 301
Set and Remove Monitoring 302
Set and Remove Monitoring Options 302
Set and Remove ALL Monitoring Options Set by a User ID or Terminal 303
Exclude Programs from Monitoring 304
Abend a Task Stopped at a Breakpoint 307
How to Produce CA InterTest for CICS Reports 308
Load a New Copy of a Program 309

Set System-Wide Options 309
Execute a Module of CNTL Commands 311
CNTL Monitoring Options 311
CNTL Monitoring Option Syntax 313
Multiple Specifications of the Same Option 314
ABI: Intercept All CICS Abends 314
ABP: Local Automatic Breakpoint Option 315
BYP: Bypass Storage Protection 315
CBP: Specify Conditional Breakpoints 316
CMD: Set Indirect Command Statements 317
CSA: Unprotect an Area in the CSA 317
CWA: Unprotect an Area in the CWA 318
DM: Data Monitoring Option 318
FEP: Set the FEP Option 318
FOL: Continue Monitoring After a Branch to Another Program 319
ICT: Instruction Counter for Preventing AICA Abends 319
KEP: Keep Data Areas in a Keep Window 320
LET: Allow a Program to Modify Storage or a Load Module 321
LNK: Set Composite Support 321
MON: Set Monitor ON Locations for Segmented Monitoring 322
MUS: Limit the Number of Times a Program Is Monitored 322
MXR: Limit the Number of CICS Requests 323
MXS: Limit Storage Usage 323
NOM: Set NO Monitor Locations for Segmented Monitoring 323
NRB: Prevent a Read Buffer Before a Breakpoint Display 324
NUP: Prevent a Program from Updating CICS Files 324
OVR: Override Error Conditions That Trigger an ABP 324
PRO: Protect Storage from Being Modified 325
RBP: Specify Request Breakpoints 325
RFC: Replace File Names 327
RNT: Prevent a Program from Modifying Its Own Code 327
RPC: Replace Program Names 327
RTD: Replace Transient Data Queue Names 328
RTS: Replace Temporary Storage Identifications 328
SDF: Structure Display Format Setting 328
SLB: Activate the Source Listing Breakpoint Facility 329
STR: Save the CICS Trace Table 329
TAL: Count How Often an Instruction Is Executed 329
TER: Change the Terminal ID 330
TON: Limit Monitoring to One Terminal 330
TRC: Statement Tracing Option 330
UBP: Specify Unconditional Breakpoints 331

USH: Unprotect Shared Storage 332
USR: Limit Monitoring to a CICS User ID 332
CNTL Capabilities 333
Monitoring 333
Monitoring Options 333
Status Display, Utility, and System Options 334
Resume Task Execution 334
Initiate Monitoring 334
Monitor Command Building Menus and Screens 335
Access the Monitoring Command Builder Main Menu 336
The Function Selection Menu 337
Monitoring 340
Unconditional Breakpoints 342
Conditional Breakpoints 346
Request Breakpoints 355
Statement Trace Options 357
Replacement Options 359
Protection Options 360
Special Options 363
Set and Remove Composite Support 365
Indirect Commands 366
Status Display 367
Utility Functions 370
Set SystemWide Options 371
Resume Task Execution 372

Chapter 10: Accessing Main Storage: CORE 375

The Main Storage Facilities (CORE) 375
Main Storage Capabilities 375
CORE Main Storage Displays 376
The Structured Format 377
The COBOL and PL/I Structured Display 377
The Assembler and CICS Areas Structured Display 379
Use Another Program's Data Definitions (USE Option) 381
Access Storage from the Menus 388
PF Key Use at a CORE Menu 389
Breakpoint-Related Areas Menu 390
Display Storage 391
Display Qualified, Indexed, or Subscripted COBOL Data Names 392
Other Ways of Changing Storage 395
Search for Data 398

Assembler Version of the Breakpoint Areas Menu 399
Examples 400
PL/I Version of the Breakpoint Areas Menu 401
System-Related Areas Menu 401
Display System-Related Areas 403
Change and Search for Data 405
CORE Commands and Advanced Options 406
Display Main Storage 407
Examples 407
Elements of a CORE Command 409
Storage Pointer 409
=WHERE directive 410
CORE Screen Scrolling Commands 411
Indirect Addressing 411
View Storage for a COBOL or PL/I Program at a Breakpoint 412
Display Qualified, Variable Length, Indexed, or Subscripted COBOL Data Names 414
View Storage for an Assembler Program at a Breakpoint 415
Use a Different Module for Assembler Structures 415
CORE Commands For PL/I Symbolic Programs 417
Display Program Code for a Program Not at a Breakpoint 419
Breakpoint Commands 419
Display Storage Related to a Task at a Breakpoint 420
Display Storage for an Inactive COBOL Program at a Breakpoint 422
Restore the Screen at a Breakpoint 423
Dynamically Acquire Main Storage at a Breakpoint 423
COBOL Like MOVE Command at a Breakpoint 424
Display Task Owned Areas 426
Set the Task Number in a CORE Command 427
Display Any CICS Control Area Structure in the Symbolic File 427
Display System-Related Areas 428
Change the Contents of Main Storage 428
The CHANGE Command (CHG) 429
Bit Manipulation 430
Verify Storage Before a Change 431
Move Data with the MOVEIN Command 431
Scroll an Address with the SET Command 432
Scan Main Storage 433
Load and Delete Programs or Displaying BMS Maps 434
Dump Main Storage 435
Chain CORE Commands 436
The CALC Option 437
Convert PL/I Statement Numbers into Displacements 438

CORE(NNNNN)=BMSG 439

Chapter 11: Accessing Auxiliary Storage: FILE 441

FILE Capabilities 441
Remote VSAM File and Temporary Storage Support 442
FILE Work Area 443
The FILE Transaction and Menus 443
Access File Facilities 443
The File Selection List 444
Initial File Display 445
Specify a File or Database 446
Dump Format 447
Character Format 449
Vertical Format 450
Structured Format 451
Sample Records in Structured Format 452
Use Structured Format When a Program Is at a Breakpoint 454
Use a Global Program Name 454
Find a Data Name 455
Universal Mode of Data Entry in the RCID, DATA, and SSA Fields 455
Access Files and Databases Protected by CA InterTest for CICS Passwords 456
Log FILE Session 456
End a FILE Session 457
PF Keys 457
Scroll Through a Record 458
Use the LOC Field to Position the Display 458
The Help Facility 460
Change Data in the Work Area 461
Save Records and Display Work Areas 464
Dump the Work Area 464
Record a Copy of the Screen Display 464
Common FILE Functions 465
VSAM Files 465
Identify the Record 466
FILE Screen Layout for VSAM Files 467
FILE Functions for VSAM Records 469
Work with DL/I Databases 477
FILE Screen Layout for DL/I Databases 478
FILE Functions for DL/I Databases 480
DB2 and SQL/DS Databases 487
Access the DB2 Facility 488

SQL Commands 488
Scroll Through the DB2 and SQL/DS Table Display 489
PF Keys 493
FILE Functions for DB2 and SQL/DS Databases 494
Redisplay SQL Commands 498
View the CICS Attachment Facility (z/OS Users Only) 498
BDAM Files 502
FILE Screen Layout for BDAM Files 503
FILE Functions for BDAM Records 504
Temporary Storage 510
FILE Screen Layout for Temporary Storage 511
FILE Functions for Temporary Storage 512
Transient Data 517
FILE Screen Layout for Transient Data 517
FILE Functions for Transient Data Records 518

Chapter 12: Using CA InterTest for CICS with DB2 and SQL/DS Programs 523

Preliminary Steps 523
Check for DB2 or SQL/DS Support 523
Compile or Assemble Programs with Symbolic Support 524
Check That the Attach Facility or Resource Adapter Is Active 524
Halt Programs Before and After Each SQL Request 526
Halt a Program Before Each SQL Request 526
Halt a Program after an SQL Request 527
Inspect Host Variables 527
SQL Return Codes 528
How to Handle SQL Error Codes 529
Advanced Debugging Techniques 532
DSNC Abends (z/OS Only) 534
The CICS Life of Task Control Block (CLOT) 535
Reason and Abend Codes--Inspecting and Interpreting 536
Inspect the Application Invocation Parameter List (RDIIN) 537
View the CICS DB2 Attachment Facility (z/OS Only) 537

Chapter 13: Analyzing Dumps Symbolically: CA SymDump for CICS Option 539

What Is CA SymDump for CICS Option? 539
Symbolic Dump Analysis 539
Manage Dumps 540
How CA SymDump Option Works with Your CICS Dump Data Set 541
Learn More About CA SymDump Option 541

Appendix A: Printing the Help File 543

Help File Printout 543
Sample MVS JCL to Print the Help File 543

Appendix B: Breakpoint Displays for NonSymbolic Programs 545

Unconditional Breakpoint 545
Automatic Breakpoint 547
Default CNTL Command 549
Location Information 549
Reason for the Breakpoint 550
Monitor Table Entry 551
Program, Transaction, and CICS Facility 552
Task Identification Number and Address 552
Program Storage Areas 553
BLL Cells (COBOL) or General Registers (NonCOBOL) 553
General Registers 554
Condition Code 554
Next and Previous Instructions 554
Data Display Area 555
Backtrace Display 556

Appendix C: Examining Dumps 559

CICS Abend Codes 560
Diagnostic Area 560
ABP and INTE Abend Error Codes 563
Execution Extents Table (Backtrace) 565
Entries in the CICS Trace Table 566
Special Entries 566
InternalProcessing Entries 568

Appendix D: Using the FOL= Option 571

Find the Program Table Entry 571
Find the Monitoring Table Entry 572

Appendix E: Monitoring Restrictions 573

Program Control Restrictions 573
Execution Restrictions 574
Special Circumstances 574

Index 575

Chapter 1: Introduction

This chapter introduces you to CA InterTest for CICS.

This section contains the following topics:

What Is CA InterTest for CICS? (see page 23)
Fast, Easy Testing (see page 24)
Stable CICS Test and Production Systems (see page 28)
Examine and Modify Main and Auxiliary Storage (see page 29)
Easy-to-Use Features (see page 29)
CA InterTest for CICS Components (see page 33)
CA SymDump for CICS (see page 33)
Take Advantage of CA InterTest for CICS (see page 34)

What Is CA InterTest for CICS?

CA InterTest for CICS is an interactive testing and debugging product for CICS applications written in COBOL, PL/1, and Assembler. Specific compilers support includes:

- Assembler
- COBOL
- COBOL II
- COBOL/370
- COBOL II or COBOL/370 with CA Optimizer/II
- IBM COBOL for MVS Release 2
- PL/1
- PL1/370
- PL/1 for MVS

CA InterTest for CICS is an essential tool for both application and system programmers:

- It makes CICS testing faster, easier, and more effective.
- It prevents CICS crashes by automatically detecting and preventing application errors before they damage CICS.
- It provides powerful capabilities for examining and modifying main and auxiliary storage.

Since CA InterTest for CICS is so easy to use, even novice CICS programmers quickly learn to take advantage of its capabilities.

Fast, Easy Testing

The primary function of CA InterTest for CICS is to help application programmers test and debug their programs quickly and efficiently. CA InterTest for CICS has many features that simplify testing, including:

- Interactive testing ability through the CA InterTest for CICS monitor
- Error detection and prevention
- Diagnostic information display screens
- Program interruption through setting breakpoints
- Isolated test sessions through monitoring by CICS User ID

Interactive Testing: The CA InterTest for CICS Monitor

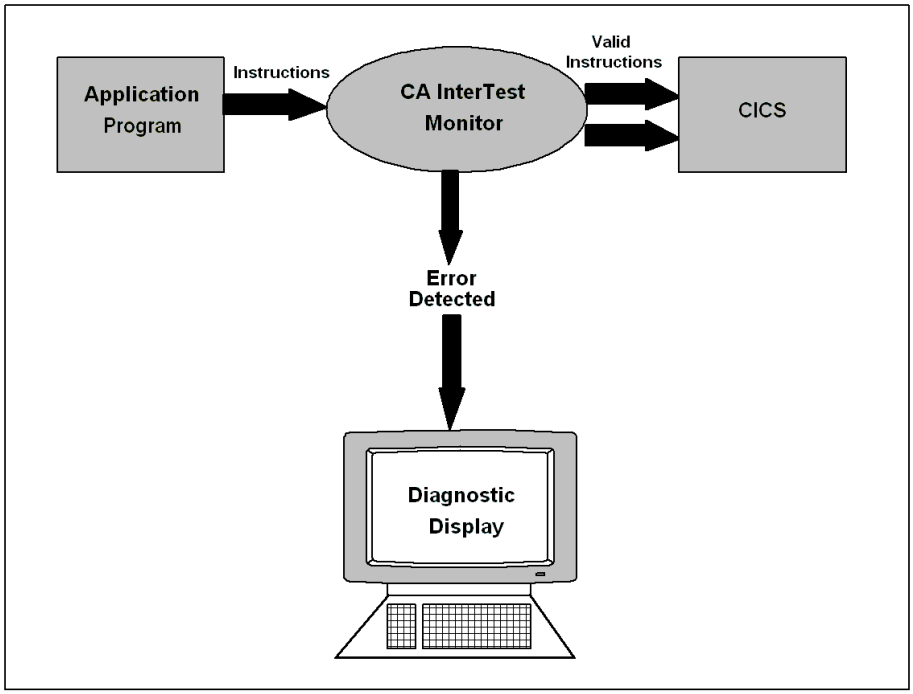
CA InterTest for CICS allows interactive testing. When CA InterTest for CICS detects an error, you can correct it dynamically (or go around it) and resume testing, allowing you to correct many errors in one test session.

With CA InterTest for CICS, you no longer need dumps. All the information you need to diagnose and correct errors is available online. You can continue testing without recompiling your program or waiting for new printouts.

CA InterTest for CICS acts as an intermediary between your CICS program and the CICS system. Here is how CA InterTest for CICS operates:

- You instruct CA InterTest for CICS to monitor a program, and then run the program.
- CA InterTest for CICS inspects every instruction and CICS command before it executes to make sure neither the program nor CICS itself will fail.
- If an error is detected, CA InterTest for CICS automatically interrupts the program. This temporary interruption in program execution is called a *breakpoint*.
- CA InterTest for CICS then displays a diagnostic screen explaining the nature of the problem and provides detailed technical information for correcting it.

  The operation of the CA InterTest for CICS monitor is shown next.



Error Detection and Prevention

CA InterTest for CICS detects and prevents errors before they occur. This means your programs will not abend. Moreover, CA InterTest for CICS does not permit storage violations that could corrupt the data of other programs or cause CICS to fail.

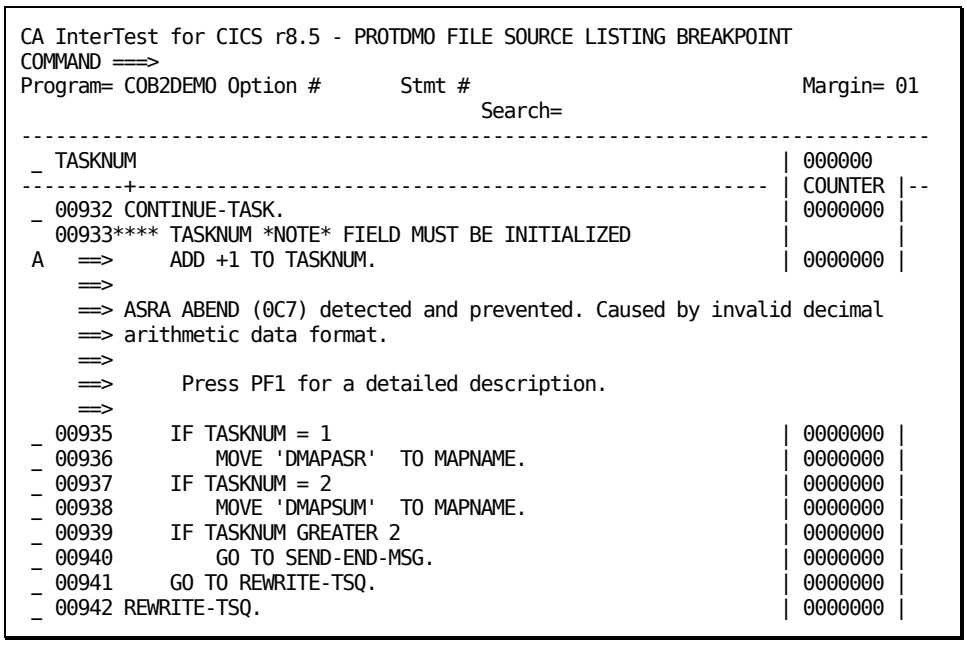
CA InterTest for CICS inspects every COBOL, PL/1 statement or Assembler instruction and every CICS Command or Macro. CA InterTest for CICS can detect and prevent the following errors:

- All storage violations (attempts to modify storage not owned by your program)
- Any CICS abend that can occur in a command level program
- All improper or invalid CICS requests (Command or Macro level)
- Any statement that would cause a program check or other abend
- All illegal or invalid instructions (for example, STOP RUN)
- All wild branches
- All violations of CICS standards

Diagnostic Information

CA InterTest for CICS provides the information you need to diagnose and correct errors. When CA InterTest for CICS detects an error, it halts the program before the error occurs. This temporary halt in program execution is called an *automatic breakpoint*. CA InterTest for CICS then displays a screen of diagnostic information indicating the statement or instruction triggering the breakpoint, and CA InterTest for CICS explains why the error occurred. Lets see what a display looks like when an error is detected.

The breakpoint display explains why CA InterTest for CICS halted the program. In the following case, CA InterTest for CICS prevented an ASRA abend caused by an invalid data format. The program instruction that triggered the breakpoint is highlighted. Now you can use all of the CA InterTest for CICS facilities to find and correct the error without analyzing a dump.



Interrupt Programs by Setting Breakpoints

To help you test and debug CICS programs effectively, you can also interrupt program execution at any point by setting breakpoints. You can set four types of breakpoints:

Unconditional— the program stops when it reaches a specified location.

Conditional—the program stops when it reaches a specified location and a prescribed condition is met (for example, a variable exceeds a certain limit).

Variable-change—the program stops at any location if the value of a specified variable has changed (COBOL and Assembler only).

Request—the program stops when it reaches specified CICS Macros and Commands, calls to DL/I or DB2, and calls to subroutines, such as database software.

You also can instruct CA InterTest for CICS to halt your program each time it executes a specified number of COBOL verbs, PL/1 statements, or Assembler instructions.

Controlling the pace at which a program executes makes it easier to pinpoint and correct logic errors. For example, when a program is halted, you can inspect the values of program variables and test data to determine whether processing is proceeding as planned. You can also dynamically change the value of a data item or generate additional test records before resuming execution.

When a program is stopped at a breakpoint, you can resume execution at any time and from any point. This means you can go around errors or dynamically alter the order in which certain routines are executed. When your program is halted at a breakpoint, you can do the following tasks:

- View your program listing and compiler output online and search for a data string
- Display data items in a Keep window to observe changes in their values
- Display and modify main storage
- Display and modify auxiliary storage
- Set and remove breakpoints
- Instruct CA InterTest for CICS to halt the program after it executes a specified number of COBOL verbs, PL/1 statements, or Assembler instructions
- Write and execute indirect commands, which are statements you insert during a test session, without recompiling the source code
- Display the path (backtrace) that brought the program to its current point
- Display the execution counts of the lines that brought the program to its current point
- Resume program execution or abend the task
- Initiate any CICS transaction

Stable CICS Test and Production Systems

CA InterTest for CICS protects CICS test and production systems because it will not permit errors that could damage CICS.

In your test CICS system, you can instruct CA InterTest for CICS to inspect every program for errors. In addition to spotting program errors, CA InterTest for CICS detects and prevents all CICS storage violations. With CA InterTest for CICS, CICS test systems are less likely to crash, so programmers can work more productively. Additionally, program errors are confined to a single program—other programs remain unaffected. Therefore, system throughput improves because trial and error testing is eliminated, test programs need fewer recompiles, and CICS is much more stable.

CA InterTest for CICS also helps maintain stable CICS production systems. Using CA InterTest for CICS in CICS test systems means programs are less likely to have bugs when they go into production. CA InterTest for CICS can also monitor new, modified, and problem programs in production until they are completely debugged. In an emergency, CA InterTest for CICS can even monitor all the programs in the production system until you isolate the error. Special options let you adapt CA InterTest for CICS to specific production situations.

Examine and Modify Main and Auxiliary Storage

CA InterTest for CICS has powerful facilities for examining and modifying main and auxiliary storage. You can take advantage of these facilities while testing a program. For example, you can interrupt program execution at various points to see how the values of program variables and test data have changed. And, you can dynamically modify storage as your testing progresses.

Example

You can halt your program, change the value in a flag, initialize a counter, or change the data in a test record, and then resume execution—all without recompiling.

You can inspect and modify main and auxiliary storage at any time—even when no program is executing and CA InterTest for CICS is otherwise inactive. System programmers can use CA InterTest for CICS to fine-tune CICS. They can display the contents of CICS control blocks and tables in main storage, making changes as necessary.

CA InterTest for CICS also makes it easy to maintain files online without writing one-time programs. You can use CA InterTest for CICS to add, update, and delete records in VSAM and BDAM files and DL/I, DB2 and SQL/DS databases. With CA InterTest for CICS, it is easy to browse a file and search for a character string. CA InterTest for CICS also lets you create and delete transient data and temporary storage records.

Easy-to-Use Features

It is easy to use the powerful CA InterTest for CICS facilities.CA InterTest for CICS is designed so that even inexperienced CICS programmers can quickly become effective users.

Getting Started with CA InterTest for CICS

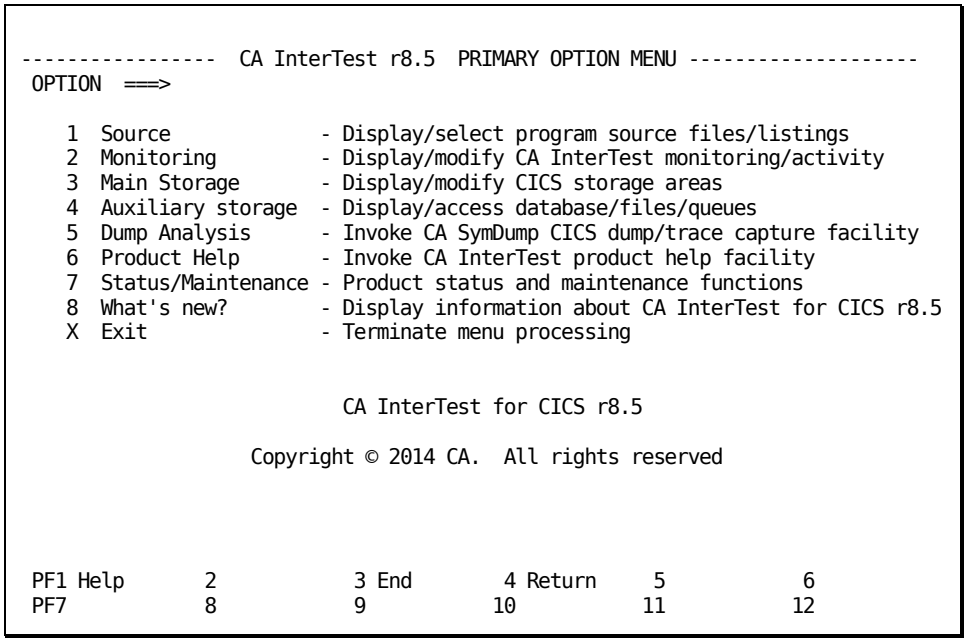
The best way to learn about CA InterTest for CICS is by using the *Primers*. COBOL, PL/1, and Assembler versions are available. The main purpose of these guides is to train new users in the basic CA InterTest for CICS features used to test and debug programs. These guides also introduce some of advanced features of CA InterTest for CICS.

We recommend that all users perform the demo sessions in the *Primers* because they provide an excellent hands-on introduction to CA InterTest for CICS.

Different Methods of Using CA InterTest for CICS

Novice CICS programmers will learn to take advantage of the many features of CA InterTest for CICS because it is not necessary to learn command syntax. Standard ISPF-like menus and selection screens make it easy to access any component in CA InterTest for CICS at any time.

The ITST transaction displays the Primary Option Menu from which you can access any CA InterTest for CICS function.



Specially formatted displays and screens also enable you to bypass menu processing by using special commands or single keystrokes for the most frequently used testing functions. For example, you can instruct CA InterTest for CICS to set monitoring, breakpoints or display main storage simply by pressing a PF key or entering a single action character next to a line in your source listing display. A command line on the top of any Source Listing or Breakpoint display lets you access a main menu or a specific menu function by entering its fastpath syntax (=X.Y.Z).

More experienced users often find it faster to enter transaction-based commands for LIST, CNTL, CORE, and FILE facilities directly.

Example

Entering LIST=COBDEMO directly from CICS immediately displays the Source Listing Display of the program COBDEMO, where you can quickly set breakpoints before you begin executing the program.

You can also mix and match different methods during a single test session to meet your testing needs.

For more information:

Menus and Displays (see page 35)

Get HELP

The CA InterTest for CICS comprehensive Help facility is available from every CA InterTest for CICS screen. Online help explains how to use all of the CA InterTest for CICS facilities, provides important information on diagnosing and correcting program errors, and includes examples. With help, all the information you need to become a productive CA InterTest for CICS user is just a PF key away.

Note: You can also access the Help facility by entering Help on a blank screen.

Once you are in the CA InterTest for CICS Help facility, you can browse through as many topics as you like. You also can print the Help screens.

Help is especially useful in helping you correct program errors. When CA InterTest for CICS halts your program at an automatic breakpoint, you can press **PF1** to get more information on the error that caused the breakpoint.

For more information:

Printing the Help File (see page 543)

User-Defined Help for Abends

You can define your own abend codes and descriptions or replace the delivered abend descriptions with your own text. If user-defined text is defined for an abend code, CA InterTest for CICS displays your site's help text instead of the delivered help text at an automatic breakpoint.

Note: Defining help for your site is discussed in the *Installation Guide.*

Online Access to Source Listings and Compiler Output

With CA InterTest for CICS, you never have to wait for printouts. CA InterTest for CICS provides online access to your source listings and compiler output, such as maps, cross-reference tables, and messages—information your online editor cannot provide. Working directly from your source listing makes it easy to set breakpoints and display main storage.

Symbolic Support

CA InterTest for CICS lets you forget about displacements or address changes after recompiling because CA InterTest for CICS lets you reference all program locations by the names you have defined in the program. You can inspect the contents of a data item simply by specifying its name. If you set breakpoints at symbolic locations and then recompile your program, CA InterTest for CICS can transfer those breakpoints to the recompiled program. You can also access all CICS system-related areas by field names.

CA InterTest for CICS even provides full symbolic support for programs that consist of separately compiled modules brought together when the program is link-edited. These modules can be written in the same or different languages.

Flexible Testing

CA InterTest for CICS makes it easy to adjust your testing plan to react to situations as they arise. For example, you can set breakpoints in your program before initiating it, and then set new breakpoints or remove existing ones after the program begins to execute. You can also create new test records during a test session if your current data is not testing your program thoroughly.

In a secure CICS region, you can take advantage of the user monitoring option to create a personal test session based only on your CICS user ID. If you specify your own User ID when setting monitoring and breakpoints, the program(s) are monitored only when you execute them, and the breakpoints and other monitoring options are directed to whatever terminal, or terminal session, you are using. The user monitoring option also permits multiple users to concurrently test the same program with different monitoring options.

CA InterTest for CICS Components

CA InterTest for CICS consists of these CICS transactions:

ITST—Displays the ISPF-like Primary Option Menu, which provides access to all other CA InterTest for CICS and CA SymDump for CICS Option facilities without having to know any other transaction or command.

LIST—Displays online COBOL, PL/1, and Assembler source listings and compiler output.

CNTL—Controls program monitoring.

CORE—Inspects and modifies main storage.

FILE—Displays and updates CICS files, DL/I, DB2, and SQL/DS databases, temporary storage, or transient data.

HELP—Provides online assistance for using CA InterTest for CICS.

You can use any of these transactions at any time. Your program does not have to be executing for you to use CORE to display or change the contents of main storage, and you can use FILE to update a file or create a test file whether or not CA InterTest for CICS is monitoring any programs.

CA SymDump for CICS

CA SymDump for CICS, an online symbolic dump facility, complements CA InterTest for CICS by helping resolve problems when dumps are necessary. This option lets you display dumps symbolically. You can then use CA InterTest for CICS to find and correct errors. CA SymDump for CICS also lets you manage the dump data set by listing, deleting, holding, and printing dumps. CA InterTest for CICS users who also have CA SymDump for CICS can access the option's facilities directly from the CA InterTest for CICS Primary Option Menu (Option 5 Dump Analysis).

Take Advantage of CA InterTest for CICS

Testing and debugging programs are critical tasks in the application development process. CA InterTest for CICS gives you the tools you need to perform these tasks quickly and efficiently. With CA InterTest for CICS, individual programs can be more thoroughly tested and made more reliable, which keeps your entire CICS system more stable. Moreover, CA InterTest for CICS provides powerful facilities for displaying and modifying main and auxiliary storage—facilities designed to meet the needs of both application and system programmers. You will also find it easy to learn how to take advantage of the many capabilities of CA InterTest for CICS.

Chapter 2: Menus and Displays

This chapter explains how to use the ISPF-like menus and selection lists to get around CA InterTest for CICS before, during, and after your test sessions. The menus:

- Are always available
- Use a standard set of PF keys
- Support a subset of ISPF commands, including the fast-path Jump (=x.y.z) command for quick navigation
- Simplify navigation among the Source Listing, Monitoring (CNTL), Main Storage (CORE), Auxiliary Storage (FILE), and CA SymDump for CICS (SYMD) displays

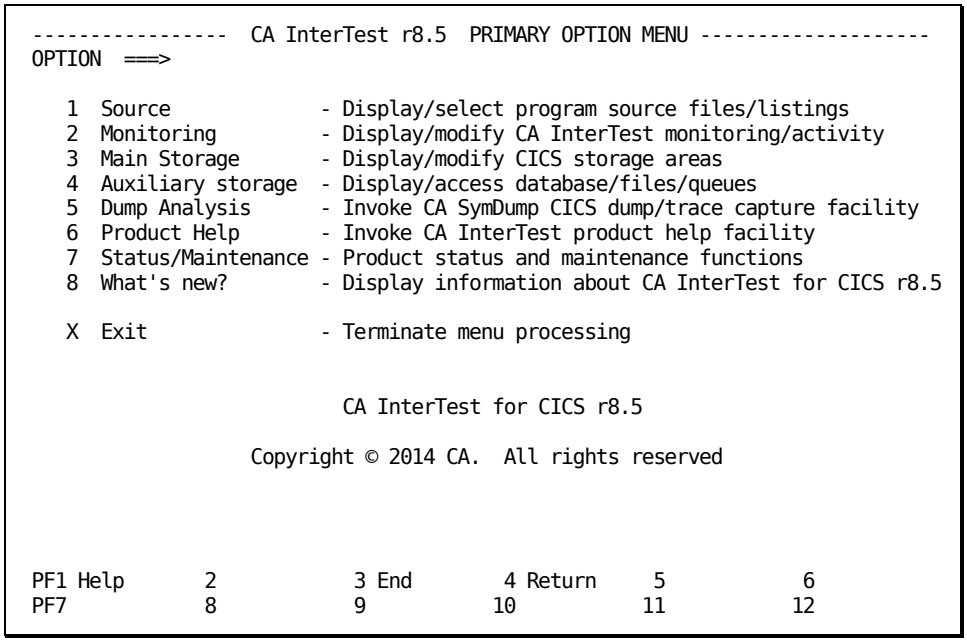
This chapter also explains how to use the Monitoring Status display, an expandable tree-like display of current monitoring entries. You will find the Status display saves testing time, since you can quickly remove any monitoring, breakpoint, or monitoring option directly from it.

Note: This chapter assumes you understand the concepts of CA InterTest for CICS monitoring and breakpoints.

The Primary Option Menu and Breakpoint Menu

Primary Option Menu

Use the CA InterTest for CICS Primary Option Menu to access and initiate any CA InterTest for CICS facility, and even CA SymDump for CICS. Notice you can also use Option 7 Status/Maintenance, to view your site's installation options, release information, and what PTFs have been applied.



Access

To access the Primary Option Menu, enter **ITST** from CICS, from any Source Listing display or breakpoint, or any CA InterTest for CICS menu.

- Before a test session, sign on to CICS, and then enter the **ITST** transaction
- From any Source Listing display or breakpoint display, type **ITST** on the command line

  Note: The command line is on line 2 of all Source displays and is available at all times.
- From the Breakpoint Primary Option Menu, type **ITST** or select option 1 Main Menu

Whats New?

Current CA InterTest for CICS users might want to browse the topics in Option 8 What's New, to read about what is new for this release.

Breakpoint Primary Option Menu

When you are at a breakpoint, CA InterTest for CICS includes a Breakpoint Primary Option Menu, which places many breakpoint activities right at your fingertips. A sample Breakpoint Primary Option Menu follows.

```
----------CA InterTest r8.5  BREAKPOINT PRIMARY OPTION MENU------------

 OPTION  ===>

    1  Main menu            Display the CA InterTest primary option menu
    2  Status               Display/remove current monitoring options
    3  Abend                Abend the breakpointed task
    4  Resume               Resume breakpointed task menu options
    5  Override             Override automatic breakpoint default processing
    6  Dump                 Cause dump, resume from next sequential instruction
    7  Disconnect           Disconnect the breakpoint from this terminal
    8  Hogan SMART          Invoke Hogan System's debugging facility
    X  Exit                 Terminate breakpoint menu processing



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

Notice that from the Breakpoint Primary Option Menu, Option 1 Main Menu takes you to the Primary Option Menu.

Access

To access the Breakpoint Primary Option Menu from any Source Listing Breakpoint, enter **MENU** on the command line or press **PF6** Menu.

Option Selection

Option Selection

Option selection for the new menus follows standard ISPF conventions. To select a menu option, enter its number in the Option==> field and press **Enter**. If you have questions about menu usage, press **PF1** for menu-specific help.

Fast-path Option Selection (=x.y.z)

The Option field of menus (and the command line of the Source Listing Facility) support the Jump (=x.y.z) command notation, also known as fastpath notation. Jump notation lets you bypass intermediate menus by making a series of selections in the format x.y.z. The equal sign indicates your selection starts from the top level menu in the current hierarchy. If you do not include the equal sign, your selection starts from the current menu.

Fast-path Notation

When a submenu to a Primary Option Menu is discussed in this guide, its fast-path from the Primary Option Menu is written in parentheses following the menu name.

Example

```
... Program Monitoring menu (2.1)...
```

the notation indicates that the Program Monitoring menu is option 2 Monitoring on the Primary Option Menu, and then option 1 Programs on the Monitoring Menu.

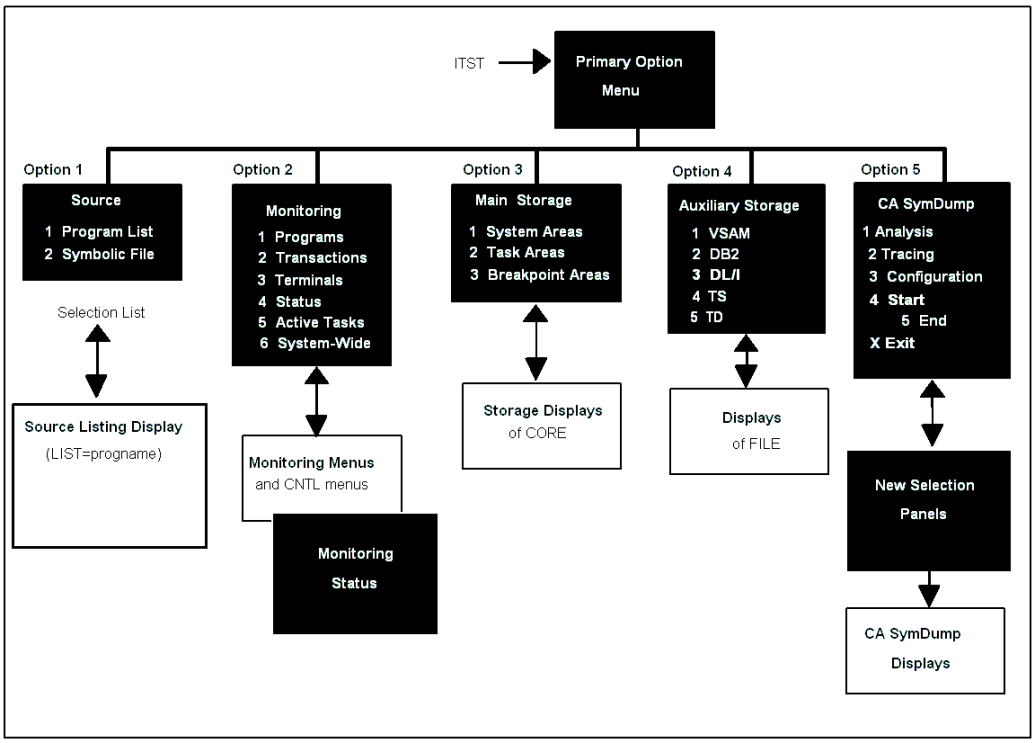
P F Keys

All menus and selection lists have a standard set of PF keys, which are listed in the following table.

PF Key	Function
PF1 Help	Displays a Help menu relating to the current menu or display. At an automatic breakpoint, gives help for the specific problem. Use Clear to exit help.
PF3 End	Ends the current display and returns to the previous menu or display.
PF4 Return	Returns to the highest level menu, either the Primary Option Menu or the Breakpoint Primary Option Menu.
PF7 Backward	Scrolls a list back one page, or scrolls back to the list item indicated by the user's cursor.
PF8 Forward	Scrolls a list forward one page, or scrolls down to the list item indicated by the user's cursor.

Menu Structure and Navigation

The menus offer easy access to the CA InterTest for CICS LIST, CNTL, CORE, FILE, and SYMD facilities. The following figure shows how the Primary Option Menus are used to access each facility.



The previous graphic does not include the last four Primary Menu Options:

6 Product Help—Displays the main menu for all help topics.

7 Status/Maintenance—**Status** provides online displays of CA InterTest for CICS and CA SymDump for CICS product components, including the genlevel, global installation options, and what maintenance has been applied. **Maintenance** allows a help administrator to maintain the optional user-help file entries of user-defined abend codes and descriptions.

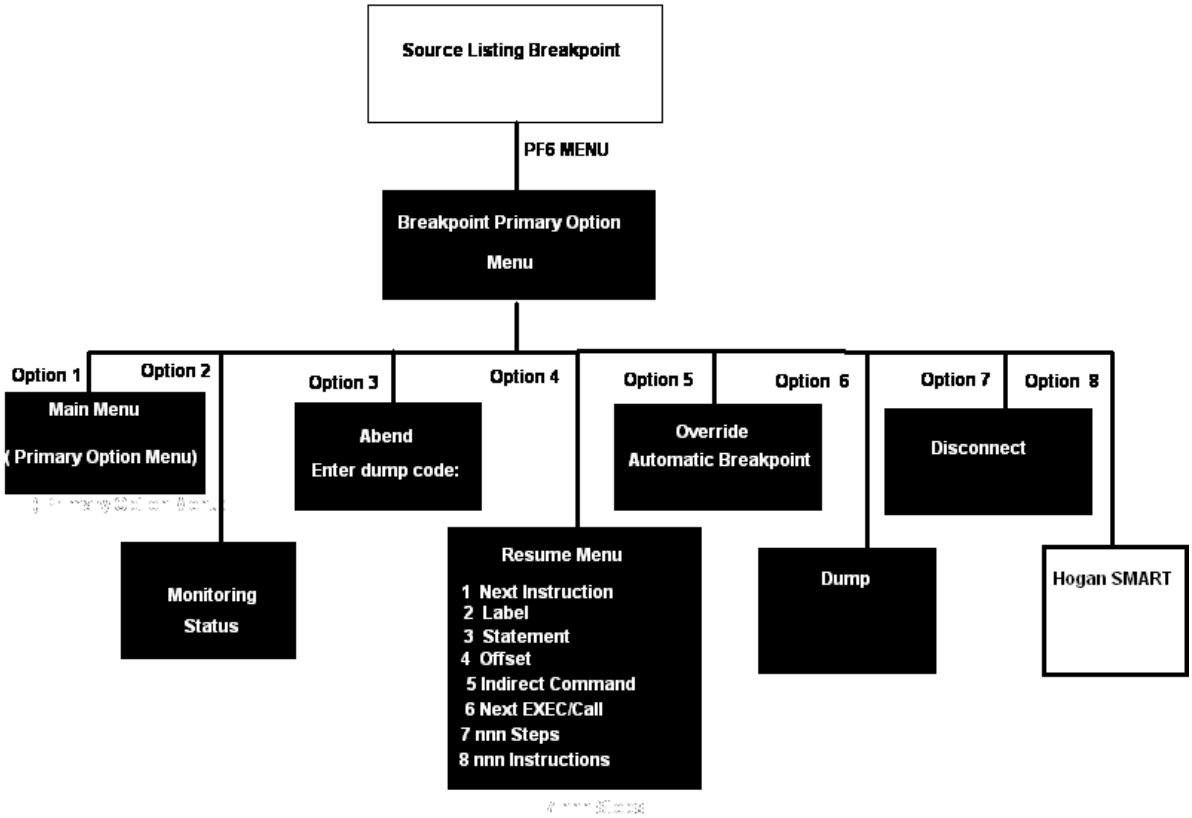
8 What's New—Displays online help topics listing the new features for this release.

X Exit—Exits menu processing. Quick exit from any menu is =**X.**

The Primary Option Menu is available at all times, even if you did not start the session from a menu. For example, if you started your CA InterTest for CICS session using LIST=COBDEMO, from the Source Listing Display you can still use PF6 Menu to display the Primary Option Menu; or if you used single-line commands to set monitoring and breakpoints for your program, and then execute the program and reach a breakpoint, PF 6 Menu displays the Breakpoint Primary Option Menu and Option 1 Main Menu accesses the Primary Option Menu.

Breakpoint Menus

The Breakpoint Primary Option Menu lets you decide how or if you want to continue execution. Notice Option 4 Resume displays a submenu with eight options for resuming execution.



Selection and Navigation Commands

CA InterTest for CICS and the CA SymDump for CICS support the following standard ISPF commands in the top OPTION==> line of menus and COMMAND ==> line of Selection Lists. These commands let you quickly navigate through screens with selection lists and tree structures.

Note: All commands do not apply to all screens.

Example

EXPAND and COLLAPSE apply only to screens with tree structure displays, and LOCATE applies only to selection lists displayed in alphanumeric order.

Commands	Function
BACKWARD BWD	Same as PF7, scrolls backwards or towards the top of the selection list. More: - (minus sign) indicates there are more selections to view using PF7 or BACKWARD.
BOTTOM BOT	Scrolls to the bottom of a selection list.
COLLAPSE	Same as PF5 Collapse. Collapses a branch in a tree structure to show only the base entry. A minus sign in front of tree structure entry indicates the branch can be collapsed.
DOWN	Same as FORWARD.
END	Same as PF3. Ends the current function and returns to the previous menu or display.
EXPAND	Expands a branch in a tree structure to show all related entries. A plus sign in front of a tree structure entry indicates the branch can be expanded.
FORWARD FWD	Same as PF8, scrolls the display list forward towards the bottom of the selection list. The More: + (plus) field indicates there are more entries to view using PF8 or FORWARD. **Note:** Use the cursor to go to an entry in the middle of a display and press PF8 (Forward) to position that entry at the top of the display.
HELP	Same as PF1, displays help for the current screen. Once in the help facility, follow the help panel instructions to access additional or related help topics. To exit help, use PF3 or Clear.
LOCATE LOC L	Locates a selection in a list based on the character string you enter. For example, LOC COBDEMO locates COBDEMO in the current selection list. Accepts masking characters * and ?

Commands	Function
MAIN	Same as PF4 Return. Returns you to the highest level menu, such as the Primary Option Menu.
REFRESH	Same as PF2 Refresh. Refreshes or redraws the current display to reflect recent modifications. For example, after removing options from a Monitoring Status display, use REFRESH to eliminate the removed entries from the tree view.
RETURN	Same as PF4 Return. Returns you to the highest level menu, such as the Primary Option Menu (before a test session) or the Breakpoint Primary Option Menu (at a breakpoint).
SELECT SEL S	Selects an option on a menu.
TERMINAL TERM	Displays the current Terminal ID.
TIME	Displays the current time.
TOP	Scrolls to the first entry in a selection list.
UP	Same as PF7, Backward.
USER	Displays the current CICS user ID.

The Monitoring Status Display

The Monitoring Status display shows you current monitoring, breakpoints and monitoring options in effect for one, or for all monitoring entries.

Sample Status

The following screen is a sample Monitoring Status display for the program COB2DEMO. This example shows the expanded monitoring entry for the Program COB2DEMO.

```
      -----------------CA InterTest for CICS r8.5  MONITORING STATUS  ---------
COMMAND ===>

  Type + to expand or  collapse option levels displayed below,
         or R to remove option(s).
                                                                          More:   +
    Option     Description               Attributes
_ COB2DEMO  Program monitor entry      COBOL II
    |       Waiting at breakpoint      Task 00043, UBP since 08:48 a.m.
_   |.ANY   User monitoring options    Active
_   |.ANY   User monitoring options    Deactivated
     |      Symbolic listing file      PROTDEM
     |      Symbolic listing file      PROTDEM
r    |RBP   Request breakpoint(s)      DSNHLI calls
     |      Option ID                  CB1E38D7
     |      From, to terminals         X508, X508
     |      On count, times thru, left 39, 0, 39
_ +  |UBP   Unconditional breakpoint   #386
_ +  |UBP   Unconditional breakpoint   #389
_    |SLB   Source listing breakpoints .ANY
_    |SLB   Source listing breakpoints X508

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

Review the following list to remember what each Option listed on this status display means.

- The COB2DEMO option identifies the program being monitored is COB2DEMO, a COBOL program. Notice the rest of the Options are indented under COB2DEMO, indicating they are options to this monitoring entry.
- The .ANY option indicates this program is being monitored under .ANY user. If you were monitoring under a single User ID, you would see a specific User ID, such as MARY01, instead of .ANY.
- Notice the rest of the options are indented under .ANY. This indicates they are options for the monitoring entry COB2DEMO when executed by .ANY user.
- The RBP option identifies a request breakpoint was set for this monitoring entry for all DSNHLI calls. Notice that the breakpoint is assigned the Option ID# CB1E38D7. This request breakpoint takes affect when the program executes at the terminal X508 and displays on the terminal X508 (the current terminal).
- The SLB option indicates the Source Listing Breakpoint display is the default for all breakpoint displays.

Access

The Monitoring Status display is conveniently accessed from the menus and the Source Listing facility.

To get a Monitoring Status for a single monitoring entry, such as a single program:

- From the Source Listing Display or Source Listing Breakpoint screen, enter **STATUS** in the COMMAND ===> line when the current program is displayed and press **Enter**, or press **PF12 Status**.
- From the Breakpoint Primary Option Menu, select option **2 Status**.

From the Primary Option Menu, complete the Monitoring Menus for the Program (2.1), Transaction (2.2), or Terminal (2.3), and select the Status option from the option list.

To get a Monitoring Status for all monitoring entries, perform the following steps:

- From the Source Listing Display or Source Listing Breakpoint screen, enter **STATUS ALL** in the COMMAND ===> line.
- From the Primary Option Menu, select Option 2.4 Monitoring Status.

The following screen shows a Monitoring Status display for all entries.

```
   -----------------CA InterTest for CICS r8.5  MONITORING STATUS
-----------------
 COMMAND ===>

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

     Option     Description               Attributes
 _ + COBDEML    Program monitor entry     COBOL
 _ + COBDEMO    Program monitor entry     COBOL
 _ + COB2DEMO   Program monitor entry     COBOL II
 _ + PL1DEMO    Program monitor entry     PL/I
                *** End of data ***




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

For more information:

Monitoring Menu Options (see page 227)

Expand and Collapse Entries

Collapse and expand entries on the monitoring status display. An option preceded by a plus sign (+) can be expanded, and an option preceded by a minus sign (-) can be collapsed.

- To collapse all entries, press **PF5** Collapse or use the **COLLAPSE** command.
- To fully expand all entries, press **PF6** Expand or use the **EXPAND** command.
- To collapse all entries below an option, tab to the selection field in front of the option, type **-** (a minus sign), and press **Enter**.
- To expand all entries for a specific option, tab to the selection field in front of the option, type + (a plus sign), and press **Enter**.

Remove Monitoring, Breakpoints, or Other Options

Enter the letter *r* in the selection field next to any option to remove it. This removes the option and any options in that branch. For example, removing monitoring for COB2DEMO removes all monitoring options and breakpoints too.

Removing breakpoints, especially request breakpoints, is simple using the status display. First, display the status report for the monitored entry and expand the monitored entry to view all options. Next, enter **r** next to the breakpoint you want to remove, and press Enter.

The following screen is an example of removing the request breakpoint for COB2DEMO, monitored under ANY user.

```
       -------CA InterTest for CICS r8.5  MONITORING STATUS -------
COMMAND ===>

  Type + to expand or  collapse option levels displayed below,
        or R to remove option(s).
                                                                    More:   +
    Option     Description               Attributes
_  COB2DEMO   Program monitor entry      COBOL II
   |          Waiting at breakpoint      Task 00043, UBP since 08:48 a.m.
_  |.ANY      User monitoring options    Active
_  |.ANY      User monitoring options    Deactivated
    |         Symbolic listing file      PROTDEM
    |         Symbolic listing file      PROTDEM
r   |RBP      Request breakpoint(s)      DSNHLI calls
                Option ID                CB1E38D7
    |         From, to terminals         X508, X508
    |         On count, times thru, left 39, 0, 39
_ + |UBP      Unconditional breakpoint   #386
_ + |UBP      Unconditional breakpoint   #389
_   |SLB      Source listing breakpoints .ANY
_   |SLB      Source listing breakpoints X508

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

Refresh the Display

After removing an option, refresh the display to reflect your changes. Use the REFRESH Command or press **PF2** Refresh.

Exit the Status Display

Press **PF3** End or **PF4** Return to exit the Status display.

Chapter 3: Source Listing Facility

The CA InterTest for CICS Source Listing facility enables you to debug your program while viewing your source code, which takes the guesswork out of testing. In addition, access all of the other CA InterTest for CICS facilities—ITST, CNTL, CORE, FILE, HELP and SYMD (for CA SymDump for CICS licensed users) directly from the Source Listing facility using menus or single-line commands.

The purpose of this chapter is to get you up and running as quickly as possible so you can use the Source Listing Facility to test your programs.

As you become familiar with the Source Listing facility, keep in mind that there is no such thing as a cookbook method of using CA InterTest for CICS. CA InterTest for CICS is a varied and powerful testing tool; how you use it depends on your particular testing needs.

Demo Sessions

In addition to reading this chapter, an excellent way to learn CA InterTest for CICS is to perform the demo sessions documented in the CA InterTest for CICS *Primer* guides. Three separate primers are available: the *Assembler Primer*, the *COBOL Primer*, and the *PL/I Primer*.

Prepare Your Program

To prepare a program for source listing testing and symbolic testing, you must modify the compile or assemble JCL to include a CA InterTest for CICS post-processor step. For source listing testing, the LISTER parameter of the post-processor is required.After modifying the JCL, recompile or reassemble the program. This puts the program's listing in the CA InterTest for CICS Symbolic File.

Note: For detailed instructions on how to modify your JCL, see the *CA Application Quality and Testing Tools Symbolic Guide*.

Start CA InterTest in Your CICS Region

CA InterTest for CICS is generally started once a day. At many sites, CA InterTest automatically starts during CICS initialization. Additionally, CA InterTest automatically starts whenever you complete a monitoring-related request from the Monitoring Menus or Submenus (Option 2 on the Primary Option Menu).

To start CA InterTest for CICS or to verify that CA InterTest for CICS is up and running, sign on to CICS and issue the following command on a Clear screen:

CNTL=START

CA InterTest for CICS responds with a message stating either that it has successfully started or that it is already active.

Note: If CICS responds with a message stating that CNTL is an invalid transaction, the systems programmer who installed CA InterTest for CICS probably changed the name of the CNTL transaction. Find out the new transaction ID and try again.

CA InterTest for CICS Transaction IDs

CA InterTest for CICS has the following standard transaction IDs: ITST, CNTL, CORE, FILE, LIST, HELP, and an additional transaction ID for licensed CA SymDump for CICS users: SYMD. These IDs might have been modified when CA InterTest for CICS was installed. If the HELP transaction ID was not modified, you can learn your current transaction IDs by entering Help on a Clear screen.

A sample CA InterTest for CICS initial screen for the HELP facility follows.

Note: Browse the Help facility by pressing Enter from the initial Help screen. To exit Help, press **PF3** or Clear**.**

```
    CA InterTest TRANSACTION CODES

CA InterTest is a productivity tool that will make your CICS testing faster,
   easier and more effective.  The following transactions are available:

  ITST  - A menu facility for all the activities included in the transactions
          below
  LIST  - To display compiler and assembler output, set and remove breakpoints
          view and modify data variables, and control program execution
  CNTL  - To set and remove breakpoints and monitoring options, control
          program execution, and report on monitoring activity
  CORE  - To display or modify main storage
  FILE  - To display or modify any CICS file or DL/I or DB2 data base
  HELP  - To obtain assistance in USING CA InterTest
  SYMD  - To display transaction dumps symbolically

  Enter any transaction code: ____ or Enter any message number: ___________

                      or Press ENTER for the HELP menu.

                         CA InterTest for CICS r8.5
               Copyright (c) 2014 CA.  All rights reserved.
```

Fill in the following table with the transaction names displayed on your screen. Use these transaction names in place of the standard CA InterTest for CICS transaction names wherever they occur in this manual.

CA InterTest for CICS Transaction IDs	Your Site's Transaction IDs
ITST	
LIST	
CNTL	
CORE	
FILE	
HELP	
SYMD	

Start a Test Session

To begin testing, be sure you have compiled or assembled your program with the CA InterTest for CICS post-processor step using the LISTER parameter, and then set monitoring and breakpoints in your program:

1. Perform CICS sign on.
2. Display your source code.
3. Set monitoring.
4. Set breakpoints.
5. Initiate your program and begin testing.

Step 1. Perform CICS Sign On

If you are working in a secure CICS region, always sign on to CICS before using CA InterTest for CICS. CA InterTest for CICS monitoring is sensitive to CICS user IDs.

Step 2. Display Your Source Code

Access the CA InterTest for CICS Source Listing facility using the following methods to view the compiled or assembled listing of your program:

- ITST Primary Menu Option access, Option **1 Source**
- Command access: **LIST=***progname*

Menu Access

Use the ITST Primary Menu, Option **1 Source**, to select your program from a selection list by program name or by SYMBOLIC file name. For details, see The Source Selection List (see page 51) later in this chapter.

Command Access

To use the single-line CICS command, sign on to CICS. On a clear screen,enter:

LIST=*progname*

progname

Specifies the name of your program.

CA InterTest for CICS displays your program on the Source Listing Display screen. From this screen set monitoring and breakpoints.

If a message informs you that the listing is not available, check whether the:

- Program name is correct
- Program is in the CA InterTest for CICS Symbolic File
- Program was compiled or assembled with the LISTER option

If there are multiple copies of the program you specified, CA InterTest for CICS asks you to select the program that you want to run. When you make your selection and press Enter, CA InterTest for CICS displays the Source Listing of your program. For details, see Maintain Synchronized Processing (see page 92), earlier in this chapter.

Step 3. Set Monitoring

While viewing the program listing, enter **monitor** in the top Command Line. This is the same as pressing **PF5 Monitor**.

Note: CA InterTest for CICS monitoring and breakpoints include a CICS user ID option. For a discussion of this option, see Set Monitoring later in this chapter.

Step 4. Set Breakpoints

Setting breakpoints enables you to control program execution. CA InterTest for CICS halts program execution when the program reaches a breakpoint, which lets you to use any CA InterTest for CICS facility before continuing execution.

Example

Set an unconditional breakpoint at the beginning of the Procedure Division, the first CSECT, or the first line of a PL/I program to gain control right from the start.

Step 5. Initiate Your Program and Begin Testing

To exit to CICS, press **PF3 End** or **Clear** to terminate the Source Listing facility. If you entered through the menus, exit the menus using **=X** or **PF3 End.**

To begin testing, execute the program as you normally would. In a secure CICS region, be sure to sign on to CICS before executing the program.

The Source Selection List

From CICS, use the ITST transaction to display the CA InterTest for CICS Primary Option Menu. Select Option **1 Source** to display the CA InterTest for CICS Source Menu.

```
 -----------CA InterTest r8.5  SOURCE MENU ----------
OPTION  ===>

   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*______ ________ ________ ________ ________ ________ ________ ________


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

This menu has two uses. Display the following items:

- Listing or a list of listings (type a **1** in the Option field).
- Symbolic file or a list of Symbolic files (type a **2** in the Option field), and then select a Symbolic file to display the Program Listings for that file.

In the unlabeled text fields, enter either the listing name or the CA InterTest for CICS Symbolic File name. Use masks to filter the selection list for items matching the filter criteria:

- Use ***** in place of a string of any length.

Example

The **c*** in the previous screen filters for all listings starting with the letter c.

- Use **+** for a single character.

Example

A mask of COBD++O filters for any seven letter item that begins with the letters COBD, has an O in the last position, and has any valid character in the fifth and sixth positions. For instance, COBDEMO, COBDXXO, and COBDABO would all meet the mask criteria, but COBDEML or COBDEM would not.

- Leave the fields blank to display **all** symbolic files or listings.

Once you specify a listing, file or mask and press **Enter**. CA InterTest for CICS displays the Source Listing Selection menu, showing all the files or listings meeting the criteria you specified on the Source Menu. Type an **S** next to the file or listing you want and press **Enter** to display that file or listing in the Source Listing display. The following screen shows a sample result of the criteria entered on the Source Menu shown previously.

```
---------CA InterTest for CICS r8.5 SOURCE LISTING SELECTION -----------
 COMMAND ===>

   Name     File        Created      Size  Attributes
 _ CICSCOBA PRTSYM1  09/20/94 16:38    29  COBOL II, no purge
 _ CICSCOBB PRTSYM1  09/20/94 16:13    21  COBOL II, no purge
 _ COBDEML  PROTSYM  07/07/94 09:32    17  COBOL, no purge, composite
 _ COBDEMO  PROTSYM  07/07/94 09:25    70  COBOL, no purge
 _ COB2DEML PROTSYM  07/07/94 09:33    25  COBOL II, no purge, composite
 _ COB2DEMO PROTSYM  07/21/94 10:34   158  COBOL II, no purge
 _ COB2DMLX PRTSYM1  09/21/94 16:10    26  COBOL II, no purge, composite
 _ COB2INSP PRTSYM1  02/01/94 09:11   155  COBOL II, no purge
 _ COB2IN25 PROTSYM  07/07/94 09:31    11  COBOL II, no purge
 _ COB2XCTL PRTSYM1  09/22/94 13:38   159  COBOL II, no purge
 _ CSBIN25  PROTSYM  07/07/94 09:29     8  COBOL, no purge
 _ C370DEML PROTSYM  07/07/94 09:33    27  COBOL/370, no purge, composite
 _ C370DEMO PROTDGG  12/12/94 12:51   161  COBOL/370, no purge
 _ C370DEMO PROTWSD  12/12/94 12:51   161  COBOL/370, no purge
 _ C370DEMO PROTSYM  07/21/94 16:35   161  COBOL/370, no purge
 _ C370IN25 PROTSYM  07/07/94 09:32    12  COBOL/370, no purge
   *** End of data ***

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

Examine the Source Listing Display Screen

A sample COBOL source listing is shown next:

```
    CA InterTest   PROTDEM FILE SOURCE LISTING DISPLAY
❶ COMMAND ==>
  Program= COBDEMO  Option #       Stmt #                          Margin= 01
  Nested=                                 Search=
❷ OPTS 1 Proc div  2 Workstor 3 Link sect  4 Dmap      5 Clst/Pmap More: 6+
       6 Data xref 7 Proc xref 8 Err msgs   9 Srch fwd  10 Srch bwd
❸ PFKS 1 Help      2           3 End        4 Profile    5 Monitor    6 Menu
       7 Backward  8 Forward   9           10           11           12 Status

--------------------------------------------------------------------------------
       _ 00001 ID DIVISION.
       _ 00002 PROGRAMID. COBDEMO.
       _ 00003 ENVIRONMENT DIVISION.
      ❹ 00004 DATA DIVISION.
       _ 00005 WORKINGSTORAGE SECTION.
       _ 00006 77  S999FIELD1             PIC S9(3).
  ❺    _00007 77  S999FIELD2             PIC S9(3)       VALUE +50.
       _ 00008 77  999FIELD1              PIC 9(3).
       _ 00009 77  999FIELD2              PIC 9(3)        VALUE 50.
       _ 00010 77  FIRSTSCREENLEN        PIC S9(4) COMP  VALUE +1696.
       _ 00011 77  MSGSCREENLEN          PIC S9(4) COMP  VALUE +1040.
       _ 00012 77  THIRDSCREENLEN        PIC S9(4) COMP  VALUE +960.
       _ 00013 77  FOURTHSCREENLEN       PIC S9(4) COMP  VALUE +1761.
       _ 00014 77  ERRORSCREENLEN        PIC S9(4) COMP  VALUE +960.
         00015*77  COMMAREALEN           PIC S9(4) COMP  VALUE +39.
```

Detailed Field Definitions

This section explains the Source Listing Display screen.

❶

Fields in which you enter commands and information:

COMMAND ===>

Enter any CA InterTest for CICS transaction, transaction-based command, or a Source Listing Display command. Enter =X.Y.Z for fast-path menu access to Primary Option Menu X.Y.Z.

Note: You cannot use this command line to enter a system-wide CNTL command.

Program=

Change the program whose code displays.

Option #

Enter an option listed on OPTS lines by number, or enter shortcut codes for CNTL functions.

Stmt #

Display a specific statement.

Nested=

Indicates the displayed COBOL nested program or the program to be searched.

Note: The COBDEMO program does **not** contain nested programs. However, if a program does contain a nested program, it displays here.

Margin=

Adjust the display margins to view code past position 80.

Search=

Instruct CA InterTest for CICS to search for a data name or string.

❷

OPTS lists entries for the Option # field. Valid options for COBOL, Assembler, and PL/I are listed in Source Listing Option # Field Entries (see page 56).

❸

PFKS describes the PF key functions.

The bottom half of the Source Listing display screen displays source code from the program named in the Program= field.

❹

Position 1 (represented by the underscore _) of each line is reserved for entering single-letter commands and for breakpoint-related functions.When a breakpoint takes effect during execution, it is identified in position 1 with one of the following letters:

A Automatic, detected by CA InterTest for CICS

U Unconditional

C Conditional

V Variable-change

R Request

❺

Displays the source code. The numbers at the left side of each line are the COBOL or PL/I statement numbers from the compiled listing, or the Assembler hexadecimal location number from the assembled listing.

❻

For COBOL and PL/I programs, there are more OPTS to view. A **More** field displays to the right of the OPTS entries. Tab to **More +** and press **Enter** to view OPTS 5 to the end. Tab to **More -** and press Enter to view OPTS 1 to 10.

Note: The display screen's format varies by terminal type. For 132-column terminals, lines of code display as is and you view them in full. For 80-column terminals, CA InterTest for CICS automatically reformats the compiler output so that you view most of the code in columns 1 to 80. Code beyond column 80 is viewed by changing the display margins in the Margin= field. For details, see Adjust the Margins (see page 75) later in this chapter.

Source Listing Option # Field Entries

The numbers listed in the following table are used in the Option # field to display specific sections of the source listing or to search for data. Using these options makes it easy to locate specific sections of your program.

Note: For PL/I programs, all of the OPTS labels cannot be displayed at once. Tab to the **More** field and press **Enter** to view options not currently displayed.

Option	COBOL	C	PL/I	Assembler
1	* Procedure Division	Data Xref	Data Name XREF	First CSECT
2	* Working-Storage	Aggegate Map	Aggregate Length Table	–
3	* Linkage Section	Storage Offset	Storage Requirements	–
4	Data Division Map	–	Static Storage Map	–
5	CLIST/Pmap	–	Variable Storage Map	Macro and Copy Source Catalog for high level output
6	Data Name XREF	Includes	Table of Offsets	Cross Reference
7	Procedure Cross Reference	Generated Code	Generated Code (Assembler-like)	Literals Cross Reference
8	Error Messages	Error Messages	Error Messages	Error Messages
9	Search Forward	Search Forward	Search Forward	Search Forward
10	Search Backward	Search Backward	Search Backward	Search Backward

Option	COBOL	C	PL/I	Assembler
11	Indirect Commands	Indirect Commands	Indirect Commands	
12	Unconditional/ Conditional Breakpoint Options	Unconditional/ Conditional Breakpoint Options	Unconditional/ Conditional Breakpoint Options	Unconditional/ Conditional Breakpoint Options
13	*Local-Storage	–	Procedure Cross Reference	–
14	–	–	Labels Cross Reference	–

***** Use these options with NESTED= entries to request a display of these sections for a specific nested program.

The sections available for display depend on which version of the post-processor LISTER parameter was used to compile or assemble the program.

Source Listing Command Line

The command line is located at line 2 of the Source Listing display so it is always available, regardless of what view options are currently in use. Use this command line to enter any CA InterTest for CICS transaction, a transaction-based line command (such as a CNTL command or CORE command), and the Source Listing Display commands listed in the following table.

Note: You cannot use this command line to enter CA InterTest for CICS system-wide CNTL options.

Command	PF Key	Description
ABEND	None	Display the Abend Breakpointed Task screen (this command is available only at a breakpoint)
ABI ON	None	Default. Turn Abend Intercept on—intercepts all CICS abends and produce an automatic breakpoint screen providing there a CICS HANDLE ABEND LABEL/PROGRAM is not active. If you want to intercept CICS abends for programs that have an active CICS HANDLE LABEL/PROGRAM then see the ABI FORCE command.

Command	PF Key	Description
ABI OFF	None	Turn Abend Intercept off—causes CA InterTest for CICS to stop intercepting CICS abends. No automatic breakpoint screen appears and the program abends.
ABI FORCE	None	Turn Abend Intercept on for all CICS ABENDS— even if a program has an outstanding HANDLE ABEND active or issues a CICS HANDLE ABEND command. Use this option when you want an automatic breakpoint screen for programs that also have active HANDLE ABENDS to process their own errors.
ADVANCE n ADV n A n	None	Advances n entries in the statement trace table and displays the resulting statement. (If n is not specified, it defaults to 1.) If data monitoring is active, past data values for the statement can also be displayed. Use this command to redisplay the execution sequence of a program. Statement tracing must be active for the ADVANCE command to function (see the TRACE option). **Note:** This command is only valid for COBOL programs.
AUTOSTEP x y AS x y	None	Resume execution x verbs at a time displaying the current source statement for a y interval of time. This command is only available at a breakpoint. One verb at a time until next CALL/COMMAND is the default. **Note:** This command is available only at a breakpoint)
BOTTOM BOT	None	GO to bottom of source
BPO	None	Jump to Breakpoint Option screen
BWD	PF7	Scrolls backward. The scroll amount is set on the Source Listing Profile.
CALLTRACE	None	Shows the program call flow leading up to the currently breakpointed program. Select programs having symbolic support from the list and display their data areas.
CHANNEL	None	Displays the currently allocated channel and the containers in the channel. From the selection list you can also display the data within a container. This feature is only valid in CICS releases that support CHANNELs and Containers.
CNTL	None	Displays the CNTL Command Menu. **Note:** The Monitoring Menu (2) is an alternative to using the CNTL menu; for details, see the chapter "Monitoring Menu Options."

Command	PF Key	Description
COMPOSITE	None	Sets composite monitoring options for monitoring of subprograms. This command is used to establish the current load module name when setting/removing breakpoint options on subprograms that are monitored under the same name in more than one composite load module. For more details, see Composite Support for Duplicate Subprograms (see page 98) at the end of this chapter.
CORE	None	Displays the Main Storage menu. Select an option to view a main storage (CORE) display.
COUNTS SHOW	None	Turns the Code Coverage COUNTER display on. To activate Code Coverage see Customizing the Source Listing Profile or issue PROFILE command.
COUNTS NOSHOW	None	Turns the Code Coverage COUNTER display off. To deactivate Code Coverage see Customizing the Source Listing Profile or issue the PROFILE command.
COUNTS RESET	None	Resets the program Code Coverage statement counters to zero.
COUNTS op value	None	Turns the Code Coverage COUNTER display on and highlights program statements that match the criteria. The source listing is repositioned to the first statement matching the criteria. Op can be: EQ or =, NE or <>, LT or <, GT or >, LE or <= or =<, GE or >= or => Valid values: 0 to 9999
COUNTS NEXT	None	Repositions the source listing to the next statement that matches the previously specified COUNT op value criteria.
COUNTS ALL	None	Removes previously specified COUNT op value criteria and redisplays source listing without special COUNTER highlighting.
DATAMON DM	None	Displays the Monitoring Command Builder menu to set and remove statement tracing and data monitoring. **Note:** This command is only valid for COBOL programs.

Command	PF Key	Description
DATAMON ON DM ON	None	Turns data monitoring on. This causes CA InterTest for CICS to capture data values for each executed statement in the program. You will then be able to redisplay these values at a breakpoint, by tracing backward and forward through the entries in the statement trace table using the PREV and ADVANCE commands. DATAMON ON forces TRACE ON. **Note:** This command is only valid for COBOL programs.
DATAMON OFF DM OFF	None	Turns data monitoring off. This causes CA InterTest for CICS to stop capturing data values for each executed statement in the program. In addition, you are no longer able to display captured data values when tracing backward and forward through the entries in the statement trace table using the PREV and ADVANCE commands. **Note:** This command is only valid for COBOL programs.
DISPLAY data-item D data-item	None	Displays the storage of a data-name. (this command is available only at a breakpoint)
DOWN	None	Shifts listing down
END	PF3	Returns to the prior display or menu.
FILE	None	Displays the Auxiliary Storage Menu. Select an option to view a file or queue.
FIND F	None	Find a string.
FP	None	Find paragraph.
FS	None	Find statement number.
FWD	PF8	Scrolls forward the amount indicated on the Source Listing Profile.
HELP	PF1	Displays help for the Source Listing Facility.
IC	None	Jump to Indirect Command screen.
ITST	See MENU	Displays the ITST Primary Option Menu.
KEEP data-item K data-item	None	Add a data-item to the static Keep window
LEFT	None	Shift listing left.

Command	PF Key	Description
LOCATE LOC L	None	Locate line number, a label, or part of the listing.
MARGIN MAR	None	Shift listing to specific margin.
MENU	PF6	Displays the high-level menu. When not at a breakpoint, it is the ITST Primary Option Menu; when at a breakpoint, it is the Breakpoint Primary Option Menu.
MONITOR	PF5	Sets monitoring for the current program using the CICS user ID displayed on the Profile.
OFFALL	None	Removes all breakpoints set by active USER or TERMINAL Active USERID is used if a valid USERID is signed on. Otherwise, the active TERMINAL is used ALL breakpoints for ALL programs/transactions/ will be removed that were set by the active USERID or active TERMINAL.
OFFALL TERM	None	Removes all breakpoints set by active TERMINAL
OFFALL USER	None	Removes all breakpoints set by active USER.
PO	None	Sets or removes storage protection options
PREV n P n	None	Backs up n entries in the statement trace table and displays the resulting statement. (If n is not specified, it defaults to 1.) If data monitoring is active, you can also display past data values for the statement. Use this command to step backward through the execution of a program. Statement tracing must be active for the PREV command to function (see the TRACE option). **Note:** This command is only valid for COBOL programs.
PROFILE	PF4	Displays Source Listing Profile, where you can change settings for the current session.
RBP	None	Sets or removes request breakpoints
RIGHT		Shifts listing right
RO	None	Sets or removes replacement options
RUN	None	Resumes execution ignoring all breakpoints. This command is available only at a breakpoint.

Command	PF Key	Description
SET	None	Initializes a data-item's value. This command is available only at a breakpoint.
SO	None	Sets or removes special options
SDFDATA	None	Set to display data in structure display format for the current program.
SDFHEX	None	Set to display data in hexadecimal/ character format for the current program.
STATUS	PF12	Displays the Monitoring Status report (Option 2.4 on the Primary Option Menu) for the current program. You can remove monitoring, breakpoints, and options directly from the Status display.
STATUS ALL	None	Displays the Monitoring Status report for all programs, transactions, and terminals CA InterTest for CICS is monitoring in the region.
STORAGE	None	Displays a listing off all USER and CICS class storage associated with the currently breakpointed task (most storage acquired by CA InterTest for CICS is not displayed), this command also validate the crumple zones for each piece of storage. See The STORAGE Facility (see page 166) in the chapter "Breakpoint Activities."
TOP	None	Go to top of source.
TRACE TR	None	Displays the Monitoring Command Builder menu to set and remove statement tracing and data monitoring. **Note:** This command is only valid for COBOL programs and is unrelated to the backtrace facility.
TRACE ON TR ON	None	Turns statement tracing on. The number of entries saved in the statement trace table is determined by the STMTTRCE option in IN25OPTS. The default is 1000. **Note:** This command is only valid for COBOL programs and is unrelated to the backtrace facility.
TRACE OFF TR OFF	None	Turns statement tracing off. TRACE OFF forces DATAMON OFF. **Note:** This command is only valid for COBOL programs and is unrelated to the backtrace facility.
UP	None	Shift listing up.
=x.y.z	None	Fast-path to Primary Menu Option x.y.z.

For more information:

Monitoring Menu Options (see page 227)
Breakpoint Activities (see page 135)

Action-Characters Supported from a Source Listing Display or Breakpoint

You can perform the following list of the functions by entering a single character on your source listing. Each of these functions is described in later chapters.

Chapter	Action Character	Function
Using Breakpoints	u	Unconditional; sets an unconditional breakpoint.
	c	Conditional; sets a conditional breakpoint.
	v	Variable-change; sets a variable-change conditional breakpoint (for COBOL or Assembler programs).

Chapter	Action Character	Function
	x	Remove; removes a breakpoint or removes a variable from the Keep window.
Breakpoint Activities	d	Display; displays program or variable storage at a breakpoint.
	k	Keep; keeps a variable in the Keep window.
	m	Modify; modifies a variable at a breakpoint.
	@	Display; displays the main storage area pointed to by a data item for a 24-bit address.
	%	Display; displays the main storage area pointed to by a data item for a 31-bit address.
Resume and Execution Options	g	Bypass; bypasses monitoring of an instruction at an automatic breakpoint.

Note: The v, d, @, %, k, and m functions should be entered next to a statement that references or defines the variable that you want to display, keep, or modify. If the statement **references** the variable, you must also **position the cursor under the variable name** before pressing Enter.

Shortcut Commands for CNTL Functions

To access a CNTL function, enter a code using the following format in the Option # field, and press Enter:

```
=nns  OR  =nnr
```

=

Required.

nn

Specifies function number (10 to 33) from the CNTL main menu.

s* or *r

*S*pecifies setting or removing the option .

Note: While the CNTL shortcut commands are supported, you will find it easier to use the Monitoring Menus, available from Primary Option Menu Option **2 Monitoring**.

The following table summarizes the Option # codes you can enter for the standard CNTL Selection Menu functions.

Option # Command	CNTL Selection Menu Function
=10s	Sets monitoring for the program displayed; same as PF5.
=10r	Removes monitoring, breakpoints, and all other options for the program displayed.
=11s and 11r	Sets and removes unconditional breakpoints.
=12s and 12r	Sets and removes conditional breakpoints.
=13s and 13r	Sets and removes request breakpoints.
=14s and 14r	Sets and removes Segmented Monitoring On option.
=15s and 15r	Sets and removes Segmented Monitoring Off option.
=16s and 16r	Sets and removes statement tracing and data monitoring.
=20s and 20r	Sets and removes Replace options.
=21s and 21r	Sets and removes Protection options.
=22s and 22r	Sets and removes Special options.
=23s	Sets Composite Support.
=24	Accesses the Indirect Commands facility.
=30	Shows the Status Display screen.
=31	Sets Utility Options screen.
=33	Resumes task execution.

For information:

Monitoring Menu Options (see page 227)
CNTL Commands and Menus (see page 295)

Source Listing PF Keys

The following table summarizes the PF key functions for the Source Listing Display screen. PF key assignments are standard across all CA InterTest for CICS menus.

Note: The ITST menus and relocated command line eliminate the need to assign CNTL, CORE, and FILE to PF keys. Navigate to these facilities using the menus, or use the command line to enter the transaction or desired single-line command.

PF Key	Function
PF1 Help	Help; displays a Help menu relating to the current Source Listing Display screen.
PF2	Unassigned.
PF3 End	End or Clear; terminates the Source Listing Facility. Displays an earlier ITST panel, if applicable, or CICS.
PF4 Profile	Displays the Source Listing Profile screen, where you can set values for your current session.
PF5 Monitor	Turns monitoring on for the program whose listing is being viewed using the CICS user ID on the Profile.
PF6 Menu	Displays the Primary Option Menu.
PF7 Backward	Scrolls the compiler output backward by one page unless the setting was changed on the Profile. Does not apply within the Keep window.
PF8 Forward	Scrolls the compiler output forward by one page unless the setting was changed on the Profile. Does not apply within the Keep window.
PF9, PF10, PF11	Unassigned.
PF12 Status	Displays the Monitoring Status for the current program. You can remove breakpoints and options directly from the Status display.

For more Information:

Breakpoint Activities (see page 135)

Search for Data

You can request CA InterTest for CICS to define, search for, and display any character string (such as a data item, label, or paragraph name). This is a quick way to go to different areas in a listing when you want to set a breakpoint or check your code.

Search for data using the Search=field, or by entering a FIND or LOCATE command on the command line.

If you know the line number that you want to view, use the line command to position the listing accordingly. For details, see View the Online Listing (see page 70) later in this chapter.

Find a Definition or Search Using the Search = Field

Use the Search= field to define or search for data as follows.

To display the definition of a data item:

1. Enter a data item name up to 31 characters in the **Search=** field.
2. Press Enter to begin the search. CA InterTest for CICS highlights the data item and its definition.

To search for a data item:

1. Enter up to 31 characters in the **Search=** field.
2. Specify the search direction in the Option # field. (Option # 9 is Search Forward; Option # 10 is Search Backward.)
3. Press **Enter** to begin the search.

For COBOL nested programs, CA InterTest for CICS first searches the current nested program (indicated in the NESTED= field) for the specified data item. If the data item is not found in the indicated program, CA InterTest for CICS then searches the main program and other nested programs for the item.

Note: It is not necessary to enter the entire data item, label, or paragraph name; enter the first few characters to start your search.

Example

If you wanted to search for TASKNUM, you could enter TASK, TASKN, and so on, in the Search= field.

Search the Program Online Listing

To find the next string within a program online listing, enter:

FIND string NEXT

String

Specifies any valid string (up to 31 characters in length). If the string contains a blank, it must be enclosed in apostrophes ('). If the string contains an apostrophe ('), it must be enclosed in quotes (").

To find the previous string within a program online listing, enter:

FIND string PREV

To repeat the find request, press **Enter**.

Locate an Area Within the Program Online Source Listing

To locate a label, line-number, offset, or special area within the program online source listing, enter:

LOCATE label | line-number | hex-offset | .xx

label

Indicates any valid label with an ASSEMBER, COBOL or PL/1 program including datanames, CSECT, procedure, and paragraph names (up to 31 characters in length).

Note: An all numeric COBOL paragraph name cannot be located; it is interpreted as a line number.

Line-number

Indicates any line number within an ASSEMBLER, COBOL or PL/1 program.

Hex-offset

Indicates any offset within an ASSEMBER program and is indicated with a leading + character.

.xx

Indicates a special indicator depending on language as follows:

COBOL:

PD = Procedure Division

WS = Working-Storage Section

LC = Local-Storage Section

LS = Linkage Section

DM = DMAP

PM = COBOL/VS PMAP/CLIST

CL = COBOL/VS PMAP/CLIST

OF = COBOL II OFFSET/LIST

LI = COBOL II OFFSET/LIST

DX = Data Name Cross Reference

.PX = Procedure Cross Reference

.EM= Error Messages

"C":

.AG = Aggregate List

.DX = Data Name Cross Reference

.EM = Error Messages

.GC = Generated Code

.IN = Includes section

.SO = Storage Offset

PL/1:

.AG = Aggregate List

.SR = Storage Registers

.SS = Static Storage

.VS = Variable Storage

.OF = Offsets

.GC = Generated Code

.PX = Procedure Cross Reference

.LX = Label Cross Reference

.DX = Data Name Cross Reference

.EM = Error Messages

ASSEMBLER:

.C1 = 1st CSECT

.1C = 1st CSECT

.MC = Macro Catalog

.XR = Cross Reference

.LI = Literals

.EM = Error Messages

View the Online Listing

Use the command line commands to position or shift the view of the listing. You can go to the top or bottom of the listing, shift the listing to the right or left a specific number of characters, or position the listing to a specific line.

Position Listing at Top

To position the listing at the top of the source code within the complete online listing, enter:

```
TOP
```

Position Listing at Bottom

To position the listing at the bottom of the source code within the complete online listing, enter:

```
BOTTOM
```

Shift Listing to the RIght

To shift the listing to the right a specific number of characters, enter:

RIGHT char-count

char-count

Indicates any valid number of characters between 1 and 49.

Default: 10

Note: The RIGHT command cannot be used on Mod5 terminals.

Shift Listing to the Left

To shift the listing to the left a specific number of characters, enter:

LEFT char-count

char-count

Indicates any valid number of characters between 1 and 49.

Default: 10

Note: The LEFT command cannot be used on Mod5 terminals.

Shift Listing Down

To shift the listing down a specific number of lines, enter:

DOWN line-count

line-count

Indicates any valid number of lines between 1 and 9999.

Default: 1

Shift Listing Up

To shift the listing up a specific number of lines, enter:

UP line-count

line-count

Indicates any valid number of lines between 1 and 9999.

Default: 1

Shift Listing to a Margin Position

To shift the listing to a specific margin position, enter:

MARGIN margin value

margin value

Indicates any valid number of characters between 1 and 50.

Note: The MARGIN command cannot be used on Mod5 terminals.

Position Listing to a Line Number

To position the listing to a specific line number, enter:

FS line-number

line-number

Indicates any line number within an ASSEMBLER, COBOL, or PL/1 program.

Position Listing to a Label

To position the listing to a specific paragraph, procedure, or label, enter:

FP label

label

Indicates any valid label within an ASSEMBLER, COBOL or PL/1 program including datanames, CSECT, procedure, and paragraph names.

Jump to the Indirect Commands Build Screen

To jump to the Indirect Commands build screen, enter:

IC

Jump to the Breakpoint Options Build Screen

To jump to the Breakpoint Options build screen, enter:

BPO

Note: The BPO command does not require any parameters. However a corresponding U/R/C must be placed in the left, line command area of the statement requiring the breakpoint option.

Scroll Forward and Backward

PF7 and PF8 scroll the compiler output backward and forward, respectively. The default scroll amount is one page; however, you can change the scroll amount on the Source Listing Profile screen, discussed in Customize the Source Listing Profile later in this chapter.

PF7 and PF8 do not apply to the Keep window, within which PF19 and PF20 perform the scroll forward and backward functions.

Change the Program

At any time during testing, CA InterTest for CICS lets you to change the program with which you are working.

Program=

To change the program displayed, follow these steps:

1. Enter the desired program name in the Program= field.
2. Press Enter to display the listing from the beginning of the program, or use the Option #, Statement #, or Search= fields to select another point of initial display before pressing Enter.

Position the Display

When you initiate the Source Listing Facility, you can easily name the program and position the display at any statement number with a single command:

LIST=progname,#nnnnn

progname

Specifies the name of the program you want to view

#

Required

nnnnn

Indicates the statement number from 1 to 99999

Note: If the requested statement number is greater than the number of statements in the listing, the greatest number is displayed.

Assembler Programs

When you initiate the Source Listing Facility you can position the Assembler display at a specific offset location in a CSECT. Use the command form:

`LIST=progname,offset`

progname

Specifies the name of the program you want to view

offset

Specifies a one to six hexadecimal number from 0 to FFFFFF

Example

To position the Source Listing display of program COBDEMO with statement 99 highlighted near the top of the display area enter:

```
LIST=COBDEMO,#99
```

Position the Source Listing display of program ASMDEMO with offset 0007E8 highlighted near the top of the display area enter:

```
LIST=ASMDEMO,7E8
```

Position the display area to include more data, source listing, or storage items using the Option #, Statement #, Margin=, and Search = fields of any Source Listing screen.

Option #

To display a specific section, select an Option # from the OPTS.

Statement #

To display a particular statement, enter the desired statement number from 1 to 9999 in the Statement # field, and press **Enter**.

Nested=

To display a particular nested program, enter the program name in the Nested = field, and press **Enter**.

Search=

To display the code defining a particular data item in Working-storage, Local-Storage, or the Linkage section, enter the data name in the Search = field, and press **Enter**.

To display the code surrounding a particular paragraph or label, enter the paragraph name or label name in the Search = field, and press **Enter.**

Displacement=

To display the listing at a specific displacement in an Assembler program, enter the displacement from 0 to FFFFFF in the Displacement= field.

Adjust the Margins

Because certain compiler output (particularly Assembler and CA Optimizer/II output) extends beyond 80 positions, it might be necessary to adjust the display margins to view more of your program's Source Listing.

Margin=

To adjust the output display to view portions of the listing to the right of position 80, perform the following steps:

1. Enter the position number of the desired left margin in the Margin= field. Valid entries are 1 through 50.
2. Press **Enter**. The screen displays the output beginning at the position specified, as shown in the following example.

Example

Assembler users often set the Margin= field to 28 as shown next.

```
                CA InterTest  PROTDEM FILE SOURCE LISTING DISPLAY
COMMAND ==>
↙
Program= ASMDEMO  Option #       Stmt #        Displacement=        Margin= 28
                                        Search=
OPTS 1 1st CSECT 2          3            4            5            6 Xref
     7 Literals  8 Err msgs  9 Srch fwd  10 Srch bwd  11           12 Bkpt opts
PFKS 1 Help      2          3 End        4 Profile    5 Monitor    6 Menu
     7 Backward  8 Forward   9           10           11           12 Status
----------+---------------------------------------------------------------------
      LOC   STMT        SOURCE STATEMENT
    000000   463+         STM   14,12,12(13)        SAVE CALLER'S REGISTERS
             464+*************************************************************

             465+*             ESTABLISH CODE ADDRESSIBILITY
             466+*************************************************************

             467+         USING *4,2,3,4,7
    000004   468+         LR    2,15
    000006   469+         LR    3,2
    000008   470+         AH    3,DFHEI4K
    00000C   471+         LR    4,3
    00000E   472+         AH    4,DFHEI4K
    000012   473+         LR    7,4
    000014   474+         AH    7,DFHEI4K
             475+*************************************************************
             476+*             OBTAIN DYNAMIC STORAGE
```

Notes:

- The statement or location numbers stay on the screen as the rest of the display shifts left or right. A plus sign (+) in the line above the display area indicates the column where shifting begins.
- If there is no output beyond position 80, CA InterTest for CICS overrides a margin setting greater than 1 and redisplays positions 1 through 80.

Customize the Source Listing Profile

CA InterTest for CICS was designed with the user in mind. The Source Listing Profile lets you set specific functions that help you meet your testing needs.

Access

To access the Profile, type **profile** on the command line and press **Enter**, or press **PF4** while viewing the Source Listing or Breakpoint screen. The Profile appears at the bottom of your screen similar to the following display:

```
CA InterTest for CICS r8.5   - PROTDEM 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           10           11           12 Status
--------+-------------------------------------------------------------------
Display titles   = R         Y (Titles), N (No titles), or R (Registers)
PF7/8 amount     = PAGE      PAGE, HALF, STOP, or a number from 1 to 9999
Step Timing      = AFTER     Stop  Before  or  After  the next verb  is executed
Stepping amount  = 001       The number of verbs   to execute
Autostepping    = OFF       ON to activate; press PF4 to change values
Source List BKPT = ON        OFF to use the detailed breakpoint display
From terminal ID = G001      Terminal ID where the program will execute
BKPT terminal ID = G001      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
```

Each Profile option has a default setting; therefore, it is not necessary to change the options in the Profile screen unless you want to change the default settings for your test session.

Display Titles, No Titles, or Registers

The default for Display Titles is Y. However, your site might have changed the default setting during installation. This setting displays the title and header lines of your Source Listing Display, including the option and PF key descriptions.

Display Titles=N

The top portion of a Source Listing Display with Display Titles = N is shown next.

```
CA InterTest  PROTDEM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= COBDEMO  Option #       Stmt #                              Margin= 01
                                        Search=
--------+-------------------------------------------------------------------
   00001 ID DIVISION.
   00002 PROGRAMID. COBDEMO.
```

To display more lines of your program's source listing instead of the titles for the Options and PF keys, perform the following steps:

1. Type **profile** on the command line and press **Enter**, or press **PF4** from the Source Listing screen to access the Profile screen.
2. Overtype the Y with **N** in the Display Titles= field.
3. Press Enter. CA InterTest for CICS replaces the option and PF key descriptions with more lines of your program's source listing.

Display Titles=R

The top portion of a Source Listing Display with Display Titles = R follows.

```
CA InterTest  PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ===>
Program= ASMDEMO  Option #       Stmt #        Displacement=        Margin= 01
                                        Search=
------------------------------------------------------------------------------
 R0R7  00394116 0038CCA0 00394028 00395028 00396028 000001D5 00000232 00397028
 R8R15 00000000 00000000 07B8CD88 0038C0C4 007B5000 0038CC38 403940F4 00000000
 Cond. Code = 0   Amode = 24
----------+-------------------------------------------------------------------
       _ 00001 ID DIVISION.
       _ 00002 PROGRAMID. COBDEMO.
...
```

To display your program's registers instead of the titles for the Options and PF keys, perform the following steps:

1. Type **profile** on the command line and press **Enter**, or press **PF4** from the Source Listing screen to access the Profile screen.
2. Overtype Y with **R** in the Display Titles= field.
3. Press Enter. When you execute your program with Display Titles=R, the title and header lines on your Source Listing Display are replaced with your program's registers and a Conditional Code line.

Note: All options and PF keys remain valid entries.

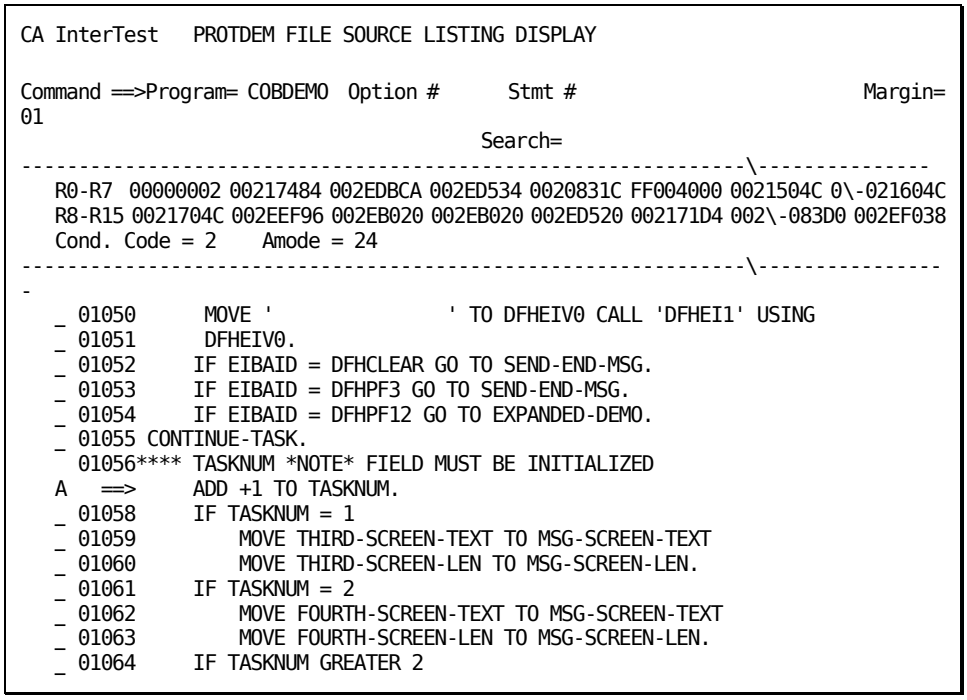
Display Titles=Y

To switch back to the Display Titles format, perform the following steps:

1. Press **PF4** to access the Source Listing Profile screen.
2. Specify Display Titles=**Y**.
3. Press **Enter**. The option and PF key descriptions redisplay.

Modifying and Locating Program Registers

When Display Titles= R, CA InterTest for CICS replaces the title lines with the contents of your program's registers, as shown in the following screen.



When registers display, the line below the registers contains:

- **Cond. Code** = reveals the current condition code and lets you know what branch is about to execute before it actually occurs.
- **Amode =** displays the current addressing mode of the program. Valid values are **24** and **31**.
- **ExecKey =** displays the current ExecKey for the breakpointed program if the CICS Storage Protection Option is active, depending on the option specified in the program definition. **User** or **CICS** are valid values. (This option is not shown in this example.)
- **TransIsolate =** displays the breakpointed task's transaction isolation option if the CICS Transaction Isolation Option is active. (This option is not shown in this example.)

Modify the program's registers by performing the following steps:

1. Overtype the displayed contents with the desired value.
2. Press **Enter**.Your changes take effect immediately.

To view the area pointed to by a register, perform the following steps:

1. Overtype the first displayed character of the register with either an at sign (@) for a 24-bit address, or with a percent sign (%), for a 31-bit address.:
2. Press **Enter**.CA InterTest for CICS displays the area pointed to by the specified register.

Scroll Amount

The scroll amount determines how much more of your source listing displays each time you use PF7 or PF8.

To set the scroll amount for scrolling backward and forward (PF7 and PF8), perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from a Source Listing screen to display the Profile screen.
2. Overtype the current PF7/8 Amount=entry with one of the following:

   PAGE

   The size (number of lines) of the display area on the Source Listing Display and Breakpoint screens.

   HALF

   Half the size of the display area.

   nnnn

   Any number of lines from 1 to 9999.

   STOP

   Go to the next or previous breakpoint in the program.
3. Press Enter. The change takes effect immediately, and you return to the Source Listing Display screen.

Notes:

- Setting the Scroll Amount to STOP is an excellent way to review all of the breakpoints set in your program
- PF7 and PF8 do not apply to the Keep window, within which PF19 and PF20 perform the forward and backward functions

Step Timing

Step timing determines where the Step or Next command stops statement execution.

To set Step timing to After, type **After** and press **Enter**.

AFTER

The program is stopped after the next verb, statement or instruction is executed.

BEFORE

The program is stopped before the next verb, statement or instruction is executed. (Default)

Step Amount

The step amount determines how many COBOL verbs, Assembler instructions, or PL/I statements will be executed at a time when you use PF10 to single-step through your program.

Stepping Amount=

To set the step amount for PF10, 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.
2. Enter the desired number of COBOL verbs, Assembler instructions, or PL/I statements from 1 to 999 in the Stepping amount= field.
3. Press **Enter**. When the Source Listing Breakpoint screen is redisplayed, the new step amount is reflected in the **PF10** (Single-step) definition, as shown in the following example.

Example

The Source Listing Breakpoint display in the following figure shows a PF10 step amount set to **004 Verbs**.

```
                  CA InterTest  PROTDEM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt  4 Profile    5 Resume     6 Menu
     7 Backward  8 Forward    9          10 004 Verbs 11 Backtrace 12 Status
------------------------------------------------------------------------------
  _ 00880 CONTINUETASK.
    00881**** 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.
      ==>
  _ 00883     IF TASKNUM = 1
  _ 00884         MOVE 'DMAPASR'  TO MAPNAME.
  _ 00885     IF TASKNUM = 2
  _ 00886         MOVE 'DMAPSUM'  TO MAPNAME.
  _ 00887     IF TASKNUM GREATER 2
  _ 00888         GO TO SENDENDMSG.
```

Note: The step value of PF10 on the Source Listing Breakpoint screen does not apply to the PF10 definition on the Detailed Breakpoint screen.

Use the Stepping Amount parameter to enable the automatic tracing feature of the Source Listing Backtrace. When the Stepping Amount is set to any number greater than one, the new value becomes the number of statements CA InterTest for CICS traces on the Source Listing Backtrace. (PF9 traces backward; PF10 traces forward).

For more information:

Breakpoint Activities (see page 135)

Turn on Automatic Single-Stepping

Automatic single-stepping lets you to go through your source listing a specified number of steps at specified time intervals.

Auto-stepping= ON

To access the Auto-stepping Parameters screen, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from any Source Listing display. The Source Listing Profile is displayed.
2. Specify Auto-stepping= **ON**.
3. Press **PF4** to activate your specification. The auto-stepping parameters are displayed.

When Auto stepping= ON, the defaults are as follows:

- Pause **one second** before repeating single-stepping or automatic tracing
- Stop at the **next CICS command, macro, or call**

The auto-stepping parameters are shown in the following screen.

```
                CA InterTest  PROTDEM 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 Det Bkpt   4 Profile    5 Resume     6 Menu
     7 Backward  8 Forward   9           10 004 Verbs 11 Backtrace 12 Status
--------------------------------------------------------------------------------

                          Autostepping parameters

Wait amount = 01    Enter the number of seconds to wait at each step (1 - 59)

Stop value  = CALL  Enter CALL to halt stepping at any COMMAND, MACRO, or CALL,
                    or the maximum number of steps to execute (1 - 9999)
```

To change the automatic stepping amount, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from the Profile screen to access the auto-stepping parameters.
2. Specify the following options:

   Wait amount

   Enter the number of seconds from 1 to 59 that CA InterTest for CICS should pause at each single-step breakpoint.

   Stop value

   Enter a value that determines when automatic single-stepping should terminate, either at a CICS command, macro, or call, or after a specified number of steps are executed.
3. Press Enter to process the new settings and return to the Profile screen.
4. Press Enter again to return to the Source Listing display or breakpoint screen.

Breakpoint Display Mode

CA InterTest for CICS has two breakpoint displays:

- The Source Listing Breakpoint version lets you to test directly from your source listing.
- The Detailed Breakpoint display requires you to use one-line CA InterTest for CICS commands and other menus.

The initial display mode depends on the setting chosen when CA InterTest for CICS was installed; however, you can change the display mode at any time by using the Profile screen. This change remains in effect until the Source List BKPT= setting is changed.

Source List BKPT=

To change the breakpoint display mode, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** to display the Profile screen. The Source List BKPT= field shows the current setting.
2. Overtype the current value with one of the following:

   ON

   Displays the Source Listing Breakpoint.

   OFF

   Displays the Detailed Breakpoint.
3. Press Enter to process the new setting and to return to the Source Listing Display screen.

Note: The default value of the Source List BKPT= field is a system-wide, CA InterTest for CICS installation option. To change the default value, have your System Administrator consult the *Installation Guide.*

From and BKPT Terminal IDs

From Terminal ID=

From Terminal ID identifies where the program executes. The default for the From Terminal ID setting depends on your system's installation option value for DFLTUSER.

- When DFLTUSER=.ANY is in effect, the default is the terminal displaying the source listing.
- When DFLTUSER=SPECIFIC is in effect and you are signed on to CICS, the default is **.ANY**. This entry tells CA InterTest for CICS to monitor the program when it runs from .ANY terminal wherever you are signed on to CICS.

To change the default From Terminal ID, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from the Source Listing display to access the Source Listing Profile screen.
2. Overtype the entry in the From Terminal ID field to specify where the program will run. Valid entries are

   terminal ID

   Runs from the specified terminal

   *****

   Runs from current terminal displaying this profile screen

.ANY

Runs from any terminal or even without a terminal

.NO

Monitors only when this program runs as a background transaction without a terminal.

Note: The BKPT terminal ID must be set to a specific 3270-type terminal that will receive the breakpoints. If From terminal ID is .N and the BKPT terminal ID is .NO or .ANY, the breakpointed task is abended with an INTE abend when the first breakpoint is encountered.

3. Press Enter to process the new From Terminal ID setting and to return to the Source Listing Display.

BKPT Terminal ID=

From Terminal ID identifies the terminal where breakpoints display. The default for the BKPT Terminal ID depends on your system's installation option value for DFLTUSER. To view your system's installation options online, use ITST Option 7.1.2.

- When DFLTUSER=.ANY is in effect, the default is the terminal that is displaying the source listing.
- When DFLTUSER=SPECIFIC is in effect and you are signed on to CICS, the default is .ANY. This entry tells CA InterTest for CICS to send the breakpoints to any terminal where the user is signed on to CICS.

To change the default, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from the Source Listing display to access the Source Listing Profile screen.
2. Overtype the entry in the BKPT Terminal ID field to specify the terminal to receive the breakpoint displays. Valid entries are:

   terminal ID

   Displays breakpoints at the specified terminal.

   *****

   Displays breakpoints at the terminal displaying this profile screen.

.ANY

For transactions executing at a 3270-type terminal, the breakpoint displays on the executing terminal.

.NO

If the transaction runs as a background task without a terminal, the transaction abends with an INTE abend when the first breakpoint is displayed.

The breakpointed task abends with an INTE abend when the first breakpoint is displayed.

3. Press Enter to process the new BKPT Terminal ID setting and to return to the Source Listing Display.

Set the User ID

CA InterTest for CICS monitoring is sensitive to user IDs. Monitor programs and set breakpoints and other monitoring options with the user ID set to .ANY, or the user ID set to a specific CICS user ID (in a secure CICS region). When the user ID is set to .ANY, everyone is monitored and the breakpoints and other monitoring options that you set take effect only if the program executes from the terminal used to set the breakpoints. When the user ID is set to a specific user ID, only that user is monitored, and the breakpoints and other monitoring options are directed to wherever the specified user is logged onto CICS.

User ID=

The default is either the current CICS user, or .ANY. To change the default, perform the following steps:

1. Type **profile** on the command line and press Enter, or press **PF4** from the Source Listing display, to access the Source Listing Profile screen.
2. Overtype the entry in the User ID field to specify the CICS user who must execute the program for monitoring and breakpoints to take effect. Valid entries are:

   specific user-ID

   Monitoring and breakpoints take effect only when the specific CICS user executes the program.This establishes a personal debugging session for the CICS user and directs breakpoints to the terminal on which the user is signed on.

   .ANY

   Monitoring and breakpoints take effect when anyone executes the program.
3. Press Enter to process the new user ID setting and to return to the Source Listing Display.

Note: If you override the default user ID setting, check the Profile before each Source Listing Session and before you set additional breakpoints during a test session.

Set the Keep Window to Display Variables in Current Statement

AUTOKEEP lets you to display items in the Keep window concerning the currently highlighted statement.

The initial setting for AUTOKEEP is chosen when CA InterTest for CICS is installed. If the IN25OPTS parameter SLBAKEEP is set to yes, which is the default, AUTOKEEP will be active. However, use the Profile screen to turn AUTOKEEP off or on again at any time.

AutoKeep Display=

To change the setting for AUTOKEEP, 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 AutoKeep Display = field shows the current setting.
2. Overtype the current value with one of the following:

   YES

   Displays in the Keep window, the variables being manipulated in the current statement, along with any data items you requested.

   NO

   Displays in the Keep window only the data items you requested.
3. Press Enter to process the new setting and to return to the Source Listing Display screen.

Set Code Coverage

Code Coverage (Counts) lets you see the number of times a verb was executed.

The initial setting for Code Counting is off, which is the default. However, use the Profile screen to turn Code Coverage off or on again at any time.

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

This feature, due to its nature, causes overhead during monitoring and should be turned off as soon as it is no longer needed.

Code Counting=

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:

   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 setting and to return to the Source Listing Display screen.

Set Structure Display Form

SDF defaults to the value set in the IN25OPTS macro, a value of HEX displays structured data in dump format and a value of DATA displays structured data in display format.

To change the current structure display format type **profile** on the command line and press Enter, or press **PF4** to access the Profile screen from the Source Listing Screen. The SDF = field shows the current setting.

Testing Activities

This section discusses testing activities for monitoring users.

Set Monitoring

A CICS user ID is assigned to all monitoring requests. Depending on how CA InterTest for CICS was installed at your site, the default may be to set monitoring for **.ANY** user, or to set monitoring for a specific CICS user. The following topics discuss how these settings affect your test session.

As long as you are working in a secure CICS region, you can override the default user ID setting using the Profile display. You should check how the user ID is set prior to setting breakpoints or monitoring for a program.

Monitor with User ID=.ANY

Setting monitoring with a user ID of **.ANY** is the same as monitoring in earlier CICS releases: the program is monitored when ANY user executes the program, and breakpoints are displayed by default on the same terminal where they were set.

Monitor with a Specific User ID

When you set monitoring for a program with a specific user ID, CA InterTest for CICS only monitors the program when it is executed by that user. In addition, breakpoints display at whatever terminal that user is signed on to. Of course, this option requires a secure CICS region and the user to be signed on to CICS when testing the program.

Monitoring with your specific user ID has the following advantages over monitoring with user ID=.ANY:

- You can change terminals and your breakpoints and monitoring options will follow you
- Your testing will not interfere with any one else testing the same program
- You can use a multi-session terminal emulator package and have breakpoints directed to whatever terminal session you are using

View and Set the User ID

To view or set the user ID that will be used for monitoring and breakpoints, access the Source Listing Profile. To access the Profile, either enter **PROFILE** on the command line and press Enter, or press **PF4**. The User ID field is at the bottom of the Profile. To change its value, delete the current entry, type in the new entry, and press **Enter**.

Also check the Terminal ID field just above the User ID field. Generally, you want Terminal ID set to .ANY when User ID is **specific**, and Terminal ID set to a specific terminal ID when User ID is .ANY.

Notes:

- Press **PF1** for field-level help.
- Press **PF4** to exit the Profile and return to the Source Listing Display.

Set Monitoring for a Displayed Program

To set monitoring for the program you are viewing, enter **MONITOR** in the command line, or press **PF5.** CA InterTest for CICS processes the request immediately using the user ID value currently set in the Source Listing Profile.

Note: If you are going to set any breakpoints, skip this step; setting a breakpoint automatically sets monitoring for the program.

Monitoring Multiple Programs

If you plan to test more than one program, you need to set monitoring and breakpoints for each program. If a program to be tested passes control to another program using XCTL or LINK, you also must set monitoring for the receiving program.

To display another program, overtype the program name in the Program= field. Specify where you want to begin the program's display by using the Option #, Stmt #, and Search = or Assembler Displacement fields. Press Enter to process your request. For Details, Change the Program (see page 73) and Position the Display (see page 73) earlier in this chapter.

With the new program displayed, repeat the steps to set monitoring and breakpoints.

Display and Search Through Nested Programs

CA InterTest for CICS provides support for COBOL II nested programs, when the COBOL nested programs recompile with the COBOL II post-compiler provided with these releases.

COBOL II nested programs let non-unique paragraph and data names to be defined across nested programs. Therefore, CA InterTest for CICS supports **qualified names** for COBOL programs. A qualified name consists of a one- to 30-byte COBOL program name, a colon, and a one- to 30-byte paragraph or data name.

Example

program1:datanam1 is a qualified name. Qualified names ensure that the correct version of datanam1, which you define in multiple programs, is displayed. Qualified names are supported by the Source Listing facility, the Breakpoint facility, CNTL commands and menus, and by the Indirect Commands facility.

When CA InterTest for CICS displays the source code for a nested program, the Nested= field appears below the Program= field. The Nested= field is 30 bytes long, and indicates the name of the nested program for the currently displayed source code. If you are using the Source Listing facility to display a COBOL nested program and you press PF8 to scroll through the entire source code of the program, the Nested= field changes each time the source code for a different nested program is displayed.

The names of all nested programs within a specific COBOL program are listed at the end of the Procedure Name Cross Reference section (Option # 7).

```
              CA InterTest  PROTDEM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= ACME2000 Option #       Stmt #                              Margin= 01
 Nested=                                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           10           11           12 Status
---------+-------------------------------------------------------------------
 _ DEFINED      PROCEDURE NAMES         REFERENCES
 _    122 3000REGISTERHANDLER          P107
 _    131 4000FORCEABEND               P108
 _    139 5000HANDLEROUTINE            D129
 _    152 9000RETURN                    P109
 _    168 9999GOBACK
 _    227 9999GOBACK                    P221
 DEFINED   CROSSREFERENCE OF PROGRAMS     REFERENCES
EXTERNAL   ACME200A . . . . . . . . . . . 120
     171   ACME200N . . . . . . . . . . . 229 114
       2   ACME2000 . . . . . . . . . . . 230
EXTERNAL   ACME2001 . . . . . . . . . . . 117
EXTERNAL   DFHEI1 . . . . . . . . . . . . 127 137 147 160 166
NESTED PROGRAM MAP
PROGRAM ATTRIBUTE CODES (RIGHTMOST COLUMN) HAVE THE FOLLOWING MEANINGS:
```

The Nested= field also indicates the nested program to which Option # 1, 2, 3, and 13 apply, and the nested program that CA InterTest for CICS searches first for any data name or paragraph name that you specify in the Search= field. If a paragraph name or data name is not found in the specified nested program, CA InterTest for CICS then searches the main program.

Example

If Option # 2 and Nested=program2 are specified, the Working Storage section for nested program program2 appears. Also, if Nested=program3 and Search=datanam1are specified, nested program program3 is searched first for datanam1; if datanam1 is **not** found, the main program is searched for datanam1.

Maintain Synchronized Processing

When CA InterTest for CICS searches the symbolic files for the COBOL and PL/I programs that you specify, it tries to match the date and time in the symbolic file to that of the load module. If a match is not found, either a Symbolic Version List or a warning message displays.

When CA InterTest for CICS searches the symbolic files for the Assembler programs you specified, the Symbolic Version List displays if multiple symbolic versions of a program are found.

COBOL and PL/I Symbolic Version Processing

When you use LIST=*program* or CNTL=ON, PROG=*program* to request a COBOL or PL/I program that has no previously declared breakpoints or monitoring options set, CA InterTest for CICS matches the **most recently compiled** symbolic file date and time to the load module date and time.

- If an exact match is found, CA InterTest for CICS automatically selects and displays it.
- If a match is not found, CA InterTest for CICS displays the Symbolic Version List from which you can either:
  - Select the appropriate symbolic file
  - Ignore symbolic processing for the program

  The Symbolic Version List screen might also explain the cause of the mismatch.

During Automatic Breakpoint processing, if a program has no previously selected symbolic file, CA InterTest for CICS matches the load module's date and time to a symbolic file.

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

Once a symbolic file is selected for a program, CA InterTest for CICS continues to use the selected file and bypasses subsequent date and time matching until all declared breakpoints and monitoring options are removed from the program, or until a CNTL=NEW,PROG=*program* command is executed. The New Copy option is also available from the Program Monitoring menu, ITST Option 2.1.

For PL/I Programs

Successful date and time matching for PL/I modules requires PL/I programs be post-compiled using the following:

- IN25SYMP from CA InterTest for CICS 5.3 or above (ESA systems) or CA InterTest for CICS 4.4 (non-ESA systems)
- The PL/I compiler installed TSTAMP=YES option, which is the default

Note: If PL/I programs were compiled with IN25SYMP from CA InterTest for CICS 5.3 or 5.4 and the program contains arrays, the program needs to be recompiled with the IN25SYMP from the 6.0 release to display PL1 array variables during a program breakpoint.

Assembler Symbolic Version Processing

Assembler programs have no internal date and time generated by the compiler, which prevents date and time matching. However, when LIST=*program* or CNTL=ON,PROG=*program* is requested for an Assembler program with multiple symbolic versions and no previously declared breakpoints or monitoring options, CA InterTest for CICS displays a Symbolic Version List from which you select the appropriate symbolic file or ignore symbolic processing for the program.

During automatic breakpoint processing, if an Assembler program has no previously selected symbolic file and has multiple symbolic file versions, CA InterTest for CICS automatically displays the **first** symbolic file in your concatenation.

Once a symbolic file is selected for an Assembler program, CA InterTest for CICS continues to use the selected file and bypasses subsequent date and time matching until all declared breakpoints and monitoring options are removed from the program, or until a CNTL=NEW,PROG=*program* command is executed.

The Symbolic Version List

The Symbolic Version List lets you select the appropriate symbolic file or exit CA InterTest for CICS and find out why a date and time mismatch occurred.

- To select a symbolic File ID, place an **S** next to the File ID.
- To exit CA InterTest for CICS without viewing any symbolic file, press **PF3**.
- To see why a mismatch occurred, consult the Comment column.

The following screen shows a sample Symbolic Version List that appears when the LIST or CNTL transactions cannot determine which symbolic file to use.

```
                CA InterTest  Symbolic Version List

❶Program = COBDEMO  ❷ Load Module Date and time =  07/07/94   9.44.13
  Loadlib = COBDEMO.loadlib                                Volume = volser
    ❸                      ❹                      ❺                      ❻
  File ID    Date      Time    Language     Comments
_ PROTDEM  07/24/94  11.44.13  COBOL      Latest Version
_ PROTTST  07/24/94  10.59.00  COBOL      Update Mode
_ PROTACC  07/23/94  12.52.00  COBOL
_ PROTPRO  07/23/94  10.01.00  COBOL




       S  Select which Symbolic file to use for program display
-------------------------------------------------------------------------
❼
PFKEYS:  1 Help      2           3 No File   4           5           6
         7           8           9          10          11          12
❽CAIN8000 The latest Symbolic version does not match the current load module
```

❶

Specifies the name of the program being processed.

❷

Displays the internal date and time from the COBOL or PL/I load module. This field is **not** displayed for Assembler programs or for PL/I programs compiled with a compiler that was installed with the TSTAMP=NO installation option.

❸

Displays the names of each version of the symbolic file containing a copy of the specified program.

❹

Lists dates and times that each version of the symbolic file was compiled.

❺

Displays the language of the compiled program.

❻

Displays information about the version listed in the File ID column, and also explains the reason for the date and time mismatch. Comments include:

Update Mode

Specifies the program is being updated by the batch post-compiler utility. It could also mean that the PROTSYM ran out of space while a post-compiler was processing this program.

Date and time Match

Specifies the program compile date and time matches the load module date and time, but it is **not** the most recently compiled version of the program.

Latest Version

Specifies the program compile date and time is the most recently compiled version of the program, but it does not match the load module date and time.

❼

Lists the PF keys available from this screen.

PF1

Accesses the Help facility.

PF3

Exit the Symbolic Version List and **ignore** symbolic processing.

❽

Can contain one of the following messages:

CAIN8000

The latest Symbolic version does not match the current load module.

Reason:

Multiple versions of the program exist, and the date and time in the current load module does not match the most recently compiled program version. If an older version of the program matches the load module's date and time, it is indicated by "Date and time Match" in Comments.

Action:

If the load module is correct, and you did not forget to do a New Copy, select the symbolic file that matches the load. If you recompiled your program and forgot to do a New Copy, press Clear to cancel the request, and then New Copy the program and retry the LIST or CNTL transaction.

CAIN8001

WARNING – SSSSSS Symbolic yydd/hh:mm mismatches load yydd/hh:mm.

Reason:

Your CA InterTest for CICS symbolic files and program load module are out of sync. This message also appears on the Source Listing Breakpoint screen when a breakpoint is about to be displayed for a COBOL or PL/I program whose symbolic file compile dates and times do not match the active load module's internal date and time.

Action:

Ensure that symbolic file members match the active load module. Causes include one of the following items:

- A program was recompiled and a New Copy was not performed.
- One or more of your symbolic files are defined incorrectly or are not available. Use VRPT transaction Option 3 to list the status of the symbolic files defined in your CICS region.

CAIN8002

WARNING – Symbolic file date and time not equal to module date and time.

Reason:

Your CA InterTest for CICS symbolic files and program load module are out of sync. This message appears when the LIST or CNTL transactions cannot find a matching load module for a COBOL or PL/I program. This message also displays on the Source Listing Breakpoint screen when a breakpoint is about to be displayed for a COBOL or PL/I program whose Symbolic file compile dates and times do not match the active load module's internal date and time.

Action:

Take the same Action specified for message CAIN8001.

CAIN8003

Invalid character entered.

Reason:

A character other than S was specified. You must use S to select a symbolic file.

CAIN8004

No selection made.

Reason:

Enter was pressed but no selection was made. Use S to select a symbolic file or press Clear.

CAIN8005

Only one "S" may be entered.

Reason:

You specified more than one S. Specify **one** S to select the symbolic file that you want to use.

Composite Support for Duplicate Subprograms

Use the COMPOSITE command to set monitoring on programs that consist of multiple subprograms that are included during link-edit, and that let you refer to the subprograms as individual programs. Use this command also to establish the current load module qualifier when a subprogram is defined, using the same monitor name, to more then one composite load module.

Given the following setup:

MAINMOD1	MAINMOD2
SUBPROG1	SUBPROG1
SUBPROG2	SUBPROG2

both MAINMOD1 and MAINMOD2 contain the same subroutines. To set unique breakpoints on the subprograms, use the following procedure.

To set unique breakpoints on subprograms

1. Use LIST=MAINMOD1 and type COMPOSITE on the command line, and set it up for composite monitoring. Switch to MAINMOD2 and do the same for it.
2. Overtype the current program name to SUBPROG1 and set a breakpoint by typing U on the desired location.

   This sets an unconditional breakpoint on SUBPROG1 of MAINMOD2 because the last COMPOSITE command during this transaction was on MAINMOD2.

3. Overtype the current program to MAINMOD1, press enter, and type COMPOSITE again.

   This establishes the current composite load modules as MAINMOD1.

4. Overtype the current program to SUBPROG1 and set a breakpoint by typing U on the desired location.

   This sets an unconditional breakpoint on SUBPROG1 of MAINMOD1.

When a source listing breakpoint occurs in a subprogram, the current load module is automatically set to the load module that owns the subprogram.

Set Breakpoints

A breakpoint halts execution of your program. CA InterTest for CICS halts the program before the instruction at a breakpoint location is executed.

The following table shows the types of breakpoints that you can set in a program:

Breakpoint	Action
Unconditional	Stops at a specified program location.
Conditional	Stops at a specified program location when a specified condition is met.
Variable-change	Stops when the value of a specified variable changes; for COBOL and Assembler only.
Request	Stops at a CICS command or macro or at calls to software.

Set breakpoints from the Source Listing Display prior to program execution, from any breakpoint during program execution, or from the Monitoring menus at any time.

For more information:

Using Breakpoints (see page 103)

Remove Breakpoints

Remove user breakpoints as quickly as they were set. In addition to the previously supported methods of removing breakpoints, use the Monitoring Status display to quickly remove any type of breakpoint.

Remove automatic breakpoints only if the error is corrected. However, you can go around the error and begin execution from another point.

For more information:

Breakpoint Activities (see page 135)
Resume and Execution Options (see page 213)
Menus and Displays (see page 35)
Using Breakpoints (see page 103)

Statement Tracing

Statement tracing causes CA InterTest for CICS to add entries to the statement trace table for each executed statement. Then view the executed statements at any breakpoint by stepping backward and forward through the trace table entries using the PREV and ADVANCE commands. Statement tracing is only valid for COBOL programs and is unrelated to the backtrace facility.

Set statement tracing from the Source Listing Display prior to program execution, from any breakpoint during program execution, or from the Monitoring menus at any time. Remove statement tracing using one of those methods. In addition, use the Monitoring Status display to quickly remove statement tracing.

For more information:

Menus and Displays (see page 35)
Breakpoint Activities (see page 135)

Data Monitoring

Data monitoring causes CA InterTest for CICS to capture data values for each statement executed in a program. The captured data values are associated with an entry in the statement trace table. Redisplay them at any breakpoint by stepping backward and forward through the trace table entries using the PREV and ADVANCE commands. Data monitoring requires that statement tracing be active and is only valid for COBOL programs.

Set or remove data monitoring from the Source Listing Display prior to program execution, from any breakpoint during program execution, or from the Monitoring menus at any time.

For more information:

Menus and Displays (see page 35)
Breakpoint Activities (see page 135)

Run Your Program

After setting monitoring and breakpoints for each program you want to test, you are ready to begin testing. To begin testing, perform the following steps:

1. Exit the Source Listing facility by pressing **PF3** or **Clear**. If you accessed a CA InterTest for CICS screen outside the Source Listing facility, you might have to press **Clear** more than once to return to CICS. CA InterTest for CICS responds with the message:

   ```
   Lister Function Terminated.
   ```

2. Initiate the program as you normally would from CICS. Depending on the breakpoints you set and the processing paths your program takes, one of the following occurs:

   - CA InterTest for CICS stops the transaction at a breakpoint (either one you set or one CA InterTest for CICS automatically triggered) because the program was about to violate a CICS coding standard.
   - The application runs to completion without being stopped by CA InterTest for CICS. The results might not be correct.

Note: If the program ends and you get incorrect results or need to test other logic paths, initiate LIST again and set additional breakpoints before retesting the program. You do not have to reset monitoring—it remains in effect until specifically removed.

Breakpoint Activities

The chapter "Breakpoint Activities" explains all of the ways you can use CA InterTest for CICS at a breakpoint.

Exit the Source Listing Facility

Type **END** on the command line and press Enter to exit the Source Listing Facility or exit the menus using **=X**.

Chapter 4: Using Breakpoints

This chapter explains how to use the Source Listing facility and the menus to set and remove breakpoints in your programs. The following bullets show the user-breakpoint types:

- Unconditional
- Conditional
- Variable-change
- Request breakpoints

The first sections of this describe how to set each type breakpoint type, giving alternative methods from the Source Listing and the menus. The later sections discuss how to remove the breakpoints.

For more information:

Menus and Displays (see page 35)
CNTL Commands and Menus (see page 295)

Set Breakpoints

A breakpoint halts execution of your program. CA InterTest for CICS halts the program at a breakpoint location before the instruction executes.

The following table shows the types of breakpoints that you can set in a program:

Breakpoint	Action
Unconditional	Stops at a specified program location.
Conditional	Stops at a specified program location when a specified condition is met.
Variable-change	Stops when the value of a specified variable changes.
Request	Stops at a CICS command or macro or at calls to software.

Setting a breakpoint automatically sets monitoring for that program, so if you are setting a breakpoint before you execute a program, you do not need to set monitoring for the program.

Where Should You Set Breakpoints?

Set breakpoints to control program execution and to pinpoint errors in logic.At a breakpoint, use CA InterTest for CICS to do the following tasks:

- Set additional breakpoints
- Execute line-by-line (single step)
- Examine or change the program path
- Display or change main storage
- Code and execute indirect commands
- Display or change auxiliary files

When Should You Set Breakpoints?

Set breakpoints before program execution or whenever the program stops at a breakpoint during execution. You do not have to set all breakpoints in advance. Set one, start executing, and then set more when you reach a breakpoint. To review your breakpoints or remove them, use the Status Display (PF12) at any time: before execution, from any breakpoint, or after your session.

Why Should You Set Breakpoints?

You might want to set a breakpoint at the first statement in the COBOL

Procedure Division, the Assembler CSECT, or PL/I program. This gives you the opportunity to set the execution pace and to determine the program path that you want to take during testing. If program control is passed from another program, examine the COMMAREA at this point.

Often, breakpoints are set at the following locations for debugging purposes:

- At paragraph names to examine variables at the start of the sections
- Before a branch to dynamically control the program path during execution
- At each location named in an EXEC CICS HANDLE CONDITION to verify error handling

An easy way to track data items during testing is to set a variable-change breakpoint on troublesome data, and then add the data item to a Keep window.During program execution, the program stops each time the variable’s value changes, and its value displays at the top of your screen in the Keep window.

For more information:

Breakpoint Activities (see page 135)

Unconditional Breakpoints

This section reviews setting unconditional breakpoints from the Source Listing and from the menus.

Set Unconditional Breakpoints from the Source Listing

While you are viewing your source listing (either before execution or during a breakpoint), set unconditional breakpoints directly in your program code, or indirectly from the Procedure Names and Cross-Reference sections of the output.

To set an unconditional breakpoint from a Source Listing or Breakpoint screen, follow these steps:

1. Enter **u** in position 1 of the line in the program listing where you want the breakpoint set, but do **not** press Enter. (You can enter more than one **u** at a time). Enter **u** next to the following:
   - A line of executable code
   - A COBOL paragraph name
   - An Assembler label
   - A PL/I label

   A breakpoint set at a paragraph name or label occurs before execution of the first verb or instruction in the paragraph.
2. Optionally, and before pressing Enter, use the Option # field and enter **12** (Bkpt Options) to access the menu for specifying breakpoint options to your unconditional breakpoints. These options let you specify:
   - If the program should stop at this breakpoint before statement execution (default) or after statement execution
   - How often the program should stop at this breakpoint (every nth time, instead of every time)
   - The terminal where breakpoints take effect
   - The terminal that receives the breakpoints
   - Indirect commands statement numbers to execute at this breakpoint
   - The user ID (or .ANY) that executes the program

3. Press **Enter** to set the breakpoints. CA InterTest for CICS flags each unconditional breakpoint with U.

   If you used Option # 12 to set Breakpoint Options, the Breakpoint Locations Menu appears. Using this menu specify your options for the breakpoint. Complete the menu. For details, see n (see page 111) later in this chapter. After completing this menu, press Enter to set the breakpoint and return to the Source Listing display or breakpoint.

The following sample screen shows how to set unconditional breakpoints.

```
             CA InterTest - PROTSYM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt  4 Profile    5 Resume     6 Menu
     7 Backward  8 Forward   9           10 001 Verb  11 Backtrace 12 Status
------------------------------------------------------------------------------
 _ 01205     PERFORM CICS-LOOP 50 TIMES.
 _ 01206     GO TO MXS-OPTION.
 u 01207 CICS-LOOP.
   01208*EXEC CICS ASKTIME
   01209*          END-EXEC.
 _ 01210      MOVE '         00849   ' TO DFHEIV0
 u 01211     CALL 'DFHEI1' USING DFHEIV0.
   01212
 u 01213 MXS-OPTION.
 _ 01214     IF  TASK-SWITCH3 EQUAL SPACE
 _ 01215         MOVE 'A' TO TASK-SWITCH3
 _ 01216         MOVE 'DMAP06' TO MAPNAME
 _ 01217         GO TO SEND-REWRITE-RETURN.
   01218*EXEC CICS HANDLE CONDITION
   01219*          NOSTG(NO-STORAGE)
```

Note: If you want to recompile your program and keep the same breakpoints in the recompiled version, set breakpoints next to paragraph names or labels, *not* lines of executable code. Then, use the CA InterTest for CICS New Program Copy function, explained in the chapter "Monitoring Menu Options," to transfer the breakpoints to the same paragraph names or labels in the recompiled program. If you set breakpoints at lines of executable code, New Program Copy transfers the breakpoints to the same *statement numbers* or *offsets* in the recompiled program, which might not be what you want.

If you compiled or assembled your program with the post-processor parameter value, LISTER=ALL, you can set unconditional breakpoints using the Cross-Reference section. This method sets breakpoints at all references to the selected data name. To use this method, follow these steps:

1. Display the Cross-Reference section using the Option # field.
2. Enter **u** in column 1 next to the data name.
3. Press Enter to set the breakpoints. CA InterTest for CICS sets the breakpoints at all references to the COBOL, Assembler, or PL/I data name. Then, CA InterTest for CICS displays the U-breakpoint indicators in the Cross-Reference section and on the lines in the Procedure Division or CSECT where the data name is referenced.

To set breakpoints at specific paragraph names, labels, or procedure names, follow these steps:

1. Display the Cross-Reference section using the Option # field; see the chapter "Source Listing Facility."
2. Enter a **u** next to the:
   - Specific paragraph name, label, or procedure name in the COBOL Cross-Reference listing
   - Symbol line in the Cross-Reference section of the Assembler listing
   - Specific label or procedure name in the Xref section of the PL/I listing
3. Press Enter. The breakpoints apply to the specific paragraph names, labels, or procedure names.

   The following screen shows how to set breakpoints at Procedure Names.

```
          CA InterTest - PROTSYM FILE SOURCE LISTING DISPLAY
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 Profile    5 Monitor    6 Menu
     7 Backward  8 Forward   9           10           11           12 Status
--------------------------------------------------------------------------------
 _ PROCEDURE NAMES                DEFN     REFERENCE
 u AFTER-REWRITE                  01328
 _ CICS-LOOP                      01207  01205
 _ CONTINUE-TASK                  00895  00824 00888
 _ DATA-NAME                      01080
 u DO-READ-VAR                    01099  01096
 _ EXPANDED-DEMO                  00971  00821 00892 00893
 _ GEN-ERR                        01289  00802 00854 01047
 _ LAST-SCREEN                    00954  00947
 _ LINK-COBDEML                   01381  01372
 u LOOP-RTN                       01204
 _ MOVE-RECORD                    01148
 _ MXR-OPTION                     01199  01372
```

To set breakpoints at all paragraph names, labels, or procedure names, follow these steps:

1. Display the Procedure Names section using the Option # field.
2. Enter a **u** next to the:
   - Procedure-Names line in the COBOL Cross-Reference listing
   - Symbol line in the Cross-Reference section of the Assembler listing
   - Declare line in the Xref section of the PL/I listing
3. Press **Enter**.
4. When prompted, press **PF3** to confirm the request for multiple breakpoints. The breakpoints apply to all paragraph names, labels, or PL/I procedures.

For more information:

Monitoring Menu Options (see page 227)
Source Listing Facility (see page 47)

Set Unconditional Breakpoints from the Menus

You can set all types of breakpoints from the Monitoring submenus of the Primary Option Menu.

To set one or more unconditional breakpoints from the menus, follow these steps:

1. Access the Primary Option Menu, and select **Option 2 Monitoring**.
2. If you already set monitoring for the program, transaction, or terminal, repeat the entry that duplicates the monitoring request. Check this by viewing the Monitoring Status display. In most cases, this is Option *1 Program*.

   If you have not set monitoring, select whether you want to monitor a program (option 1), transaction (option 2), or terminal (option 3). Most users monitor the program.
3. On the Program, Transaction, or Terminal Monitoring Menu:
   - Complete the top part of the Monitoring menu to specify (or duplicate) how the program, terminal, or transaction will be monitored.
   - Complete the bottom part of the menu by entering s next to the UBP option for Unconditional Breakpoints.
   - Press Enter to process your request.

For more information:

Menus and Displays (see page 35)

Examples

The following screen shows the entries on the Program Monitoring (2.1) Menu to set an unconditional breakpoint for the program COBDEMO being monitored by user ID BARNEY1.

```
------------ CA InterTest r8.5   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  . .barney1_     User (or .ANY) for whom the program is monitored

                                                                      More:  +
   Option      Description
 _ Status     Display and/or remove monitoring options (S only)
 _ Monitor    Monitoring (R removes monitoring and all options previously set)
 s UBP        Unconditional breakpoints (specific program only)
 _ CBP        Conditional breakpoints (specific program only)
 _ RBP        Breakpoints for CICS, DB2, DL/I or external CALL requests
.
.
.

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

If you select UBP from the Monitoring submenu, the Breakpoint Locations screen appears. COBOL, PL/I, and Assembler versions of the Breakpoint Locations screen are available. The following screen is the COBOL version, which shows Set Breakpoint Locations for a COBOL Program at statement #1

```
CA InterTest MONITORING COMMAND BUILDER  COBOL BREAKPOINT LOCATIONS  11

    SET breakpoint locations for PROG=COBDEMO  in any of the following fields:
    (For qualification by nested program name, specify 'ProgramID:Label')
  ❶
  Para/Data _____________________________       _____________________________
     Names: _____________________________       _____________________________
            _____________________________       _____________________________
  Statement
❷ Numbers: 1___    _____    _____    _____    _____    _____    _____    _____

❸ Offsets: ____    _____    _____    _____    _____    _____    _____    _____

            All paragraph names:  _  ❹                          After: _❺

           Enter 'n' to stop only every n'th time:                        ____❻
           Term ID (or .ANY or .NO) where breakpoints will take effect:   ____❼
           Term ID (or .ANY) that will receive the breakpoints:           ____❽
           Statement no. of indirect command(s) to be executed:          ____ ❾
           user ID (or .ANY) who will execute the program:             ❿.BARNEY1



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

Specify Breakpoint Locations

To set breakpoint locations, enter the locations in fields ❶ through ❸. You can specify up to nine breakpoints.

- For a COBOL program, this can be a paragraph name or data name ❶ , a statement number ❷, or the hexadecimal displacement (offset) from the beginning of the program ❸.
- For an Assembler program, breakpoint locations can be Assembler labels, data names, or offsets.
- For a PL/I program, breakpoint locations can be offsets or, if you have the PL/I symbolic option, PL/I labels or statement numbers. PL/I labels can be qualified by Procedure Name using the syntax: Procedure-Name:Label.

- Use the following rules when specifying information:
- If you specify statement # 1, CA InterTest for CICS sets a breakpoint at the first executable instruction in a COBOL Procedure Division, Assembler CSECT, or PL/I procedure.
- COBOL statements can contain more than one verb. In specifying statement numbers, indicate the verb at which the breakpoint should occur. A statement number of nnn or nnn indicates the first verb in the statement, *nnn*.1 indicates the second verb, *nnn*.2 indicates the third verb, and so on. For example, a statement number entered as 503.3 specifies a breakpoint at the fourth verb in statement number 503.
- If you specify a COBOL, Assembler, or PL/I data name, breakpoints are set at every reference to that data name.
- Enter an **X** in the All paragraph names field (COBOL) ❹, All Assembler labels field, All Labels field (PL/I), or All Procedure Names field (PL/I) to set breakpoints at all of the indicated locations.
- Enter an **X i**n the After field❺ to stop after the statement executes.

Specify Breakpoint Options

The following specifications are optional:

- If you set a breakpoint within a loop, specify in field ❻ how often the program should be halted.

Example

If you specify 3, the breakpoint occurs every third time the program passes through the loop.

- Specify in field ❼ the ID of the terminal where the program must be executing for breakpoints to take effect. If you leave this field blank, it defaults to your current terminal. Enter .**ANY** to have breakpoints take effect at all terminals, even when the program executes without a terminal. Enter .**NO** to have breakpoints take effect only when the program executes without a terminal.
- Specify in field ❽ the ID of the terminal to receive breakpoint displays. If you leave this field blank, it defaults to your current terminal. Enter .**ANY** to have breakpoint screens displayed at the terminal where the program is executing when the breakpoints occur.
- Specify in field ❾ the statement number of the indirect command that you want to invoke at the breakpoint you have set.

- Specify in field ⓾ the CICS ID of the user who must execute the program for the breakpoint to take effect. Valid entries are a specific CICS user ID or **.ANY**. The default varies depending on your installation option settings for DFLTUSER.
- When you have entered all the necessary information, press **Enter**.

Example

The following example shows how to set unconditional breakpoints at four locations in the COBOL program COBDEMO.

```
     CA InterTest MONITORING COMMAND BUILDER   COBOL BREAKPOINT LOCATIONS  11


   SET breakpoint locations for PROG=COBDEMO  in any of the following fields:

                          ↙
 Para/Data return transid_________________      ________________________________
    Names: _______________________________      ________________________________
           _______________________________      ________________________________
 Statement  ↙          ↙          ↙
  Numbers: 1____    136__    174.2    _____    _____    _____    _____    _____


   Offsets: _____    _____    _____    _____    _____    _____    _____    ____

            All paragraph names:  _                                    After:_

            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: BARNEY1



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

The information in the previous example causes breakpoints to occur at:

- Paragraph RETURN-TRANSID
- Statement 1, the beginning of the Procedure Division
- The first verb in statement 136
- The third verb in statement 174

Because the user ID field is specified as BARNEY1 and the terminal identification fields are left blank, these breakpoints take effect only if COBDEMO is executed by the user ID BARNEY1. The breakpoints display at whatever terminal BARNEY1 is signed on under CICS.

For more information:

Breakpoint Activities (see page 135)
Monitoring Menu Options (see page 227)

Conditional Breakpoints

This section reviews setting conditional breakpoints from the Source Listing and from the Menu.

Set Conditional Breakpoints from the Source Listing

To set conditional breakpoints, specify the location and then the condition. Following these steps, specify one or more locations directly from the Source Listing.

1. Enter **c** in column 1 of any line in:
   - Your program listing (next to a paragraph, label, or executable code)
   - The Procedure Names section of the COBOL Cross-Reference listing (for all paragraphs or selected paragraphs)
   - The Cross-Reference section of COBOL, Assembler, or PL/I programs (next to a data name, label, or all labels)
   - If you enter **c** next to a data name in the Cross-Reference section, the breakpoint takes effect at all locations where the item is referenced.
   - If you enter **c** next to the Procedure-Names line in the Procedure names section of the COBOL Cross-Reference listing or next to the Symbol line in the Cross-Reference section of the Assembler listing, the breakpoints takes effect at **all** paragraph names or labels.

     Note: You can specify more than one **c** before pressing Enter.
2. Optionally, select **Option # 12** Breakpoint Options to set options for your conditional breakpoints.
3. Press **Enter**.

Another way of specifying the same condition for multiple locations in your program is to use the ITST menus to specify both the locations and the conditions. For details, see Set Conditional Breakpoints from the Menus (see page 115) later in this chapter.

To set conditional breakpoints that apply to multiple locations in a program, follow these steps:

1. Enter a **c** in column 1 of *each* line in your source code at which you want to set a conditional breakpoint.
2. Press Enter. CA InterTest for CICS displays the first conditional breakpoint locations screen.
3. Specify one or more conditional breakpoint locations. CA InterTest for CICS displays a Conditional Breakpoint menu.

The following is a simplified COBOL Conditional Breakpoint entry screen. In this example, the breakpoint takes effect only if the condition If PREV-ACCT equals NEXT-ACCT is met. For additional instructions on completing this screen, see Setting and Removing Conditional Breakpoints from the Menus.

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

Enter LEFT SIDE
    Data Name TASK-TEXT_____________________________________________________


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

Enter RIGHT SIDE
    Data Name _______________________________________________________________
       OR
    Literal C'ACCOUNT NOT FOUND'____________________________________________




_ ENTER  S  to drop monitoring on a true condition



Press PF9 to go to complex conditional screen if necessary
PF1 Help       2           3 End        4 Return     5           6
PF7            8           9 Complex   10           11          12
```

The For Location field indicates the statement number with the source code at which you specified **c** on the Source Listing display. Typing an S in the drop monitoring on a true condition field causes monitoring to be dropped for the rest of the current task when the conditional breakpoint is true.

Note: The For Location field does not display when you specify the location using the menus.

Press **Enter** after completing the Conditional Breakpoint menu. CA InterTest for CICS sets the conditional breakpoints and returns you to the Source Listing display.

Notes:

- Conditional Breakpoint menus for Assembler and PL/I programs and the complex Conditional Breakpoint menu for COBOL programs have additional entry fields.
- To replace an unconditional breakpoint with a conditional breakpoint, you must first remove the **u** before specifying the **c**. To remove the **u,** overtype it with **x,** and press Enter. To place a conditional breakpoint at that location, enter **c** where the **u** was, and press Enter.

Set Conditional Breakpoints from the Menus

Conditional breakpoints are set from the same Monitoring submenus used to set monitoring and unconditional breakpoints. Follow the same steps given in Set Conditional Breakpoints from the Menus (see page 115). However, instead of entering s next to the UBP option, enter **s** next to the CBP option for conditional breakpoints, as shown in the next screen.

```
------------- CA InterTest r8.5   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  . .barney1_     User (or .ANY) for whom the program is monitored

                                                                    More:  +
   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)
 s CBP          Conditional breakpoints (specific program only)
 _ RBP         Breakpoints for CICS, DB2, DL/I or external CALL requests
.
.
.

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

Setting Conditional Breakpoints for COBDEMO, monitored for BARNEY1

First, CA InterTest for CICS displays the Conditional Locations screen shown next.

```
CA InterTest MONITORING COMMAND BUILDER  COBOL CONDITIONAL LOCATIONS 12

    SET breakpoint locations for PROG=COBDEMO  in any of the following fields:

  Para/Data ________________________________     ________________________________
     Names: ________________________________     ________________________________
            ________________________________     ________________________________
  Statement
   Numbers: _____    _____    _____    _____    _____    _____    _____    _____

    Offsets: _____    _____    _____    _____    _____    _____    _____    _____
                                                                              ↙
             All paragraph names:  _             All instructions: x    After:_

             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:              .BARNEY1


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

This screen is similar to the screen used to specify unconditional breakpoints, and you should enter the information in the same way. For details, see Set Unconditional Breakpoints from the Menus (see page 108) later in this chapter.

One special feature lets you set a conditional breakpoint at *all* instructions. Enter an **x** in the All Instructions field to request this option, as shown in the previous example. If you request the All Instructions option and set a variable-change breakpoint, the breakpoint takes effect whenever the value of the variable changes.

After specifying the breakpoint locations, press Enter to display the Conditional Breakpoint screen. On this screen specify the condition that must be satisfied for the breakpoint to take effect.

There are two versions of the Conditional Breakpoint screen: a simplified version for COBOL users, and a detailed version for PL/I and Assembler users and for those COBOL users who prefer this version. Using either version you specify the condition in this format:

```
left side relational operator right side
```

If you are a COBOL user, the following screen appears.

```
 CA InterTest MONITORING COMMAND BUILDER  CONDITIONAL BREAKPOINT
❶ Enter LEFT SIDE
       Data Name  taskswitch_____________________________________________


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

   Enter RIGHT SIDE
       Data Name ____________________________________________________________
          OR
❸     Literal 3_____________________________________________________________

          Press PF9 to go to complex conditional screen if necessary


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

Field ❶ specifies the left side of the comparison, which is a data name.

Field ❷ specifies the relational operator:

EQ for equal

NE for not equal

GT for greater than

LT for less than

GE for greater than or equal to

LE for less than or equal to

Field ❸ specifies the right side of the comparison, which can be a data name, literal or figurative constant.

Variable-Change Breakpoints

COBOL and Assembler users set a conditional breakpoint to take effect whenever the value of a variable changes. To do this, type **x** in the All Instructions field on the Conditional Locations screen.

COBOL users must specify the variable name in fields ❶ and ❸ on the Conditional Breakpoint screen. The condition for a variable-change breakpoint is:

```
variableA NE variableA
```

Assembler users must specify the variable name in fields ❶ and ❺ on the Detailed Conditional Breakpoint screen, as discussed in The Detailed Conditional Breakpoint Screen (see page 121) later in this chapter.

The operator entered in field ❷ determines whether the breakpoint occurs whenever the value of the variable changes (NE), or only when the value increases, decreases, or does not change. Whenever you re-execute the program, the initial value of the variable resets to its value at the beginning of program execution.

Literal Formats

Specify four types of literals for the right side of the comparison:

Character Specify character literals for comparisons with any field except a COBOL data name defined as COMP or COMP-3. Enter the literal as **C'character data' , X' hexadecimal data'**, or a combination of both. For example, C'ABC', Xv0102',or C'ABC'X'0102'.

Packed	Specify packed decimal literals for comparisons only with COBOL data names defined as COMP-3. Enter the literal as **P'number'**. The number can be preceded by a minus sign. For example, P'123' or P'-123'.
Halfword	Specify halfword literals for comparisons with COBOL data names defined as COMP with a length of four, as in S9(4) COMP. Enter the literal as **H'number'**. The number can be preceded by a minus sign. For example, H'12' or H'-12'
Fullword	Specify fullword literals for comparisons with COBOL data items defined as COMP with a length of eight, as in S9(8) COMP. Enter the literal as **F'num'**100' or F' -100'.

Figurative Constants

Specify the following figurative constants for the right side of the comparison:

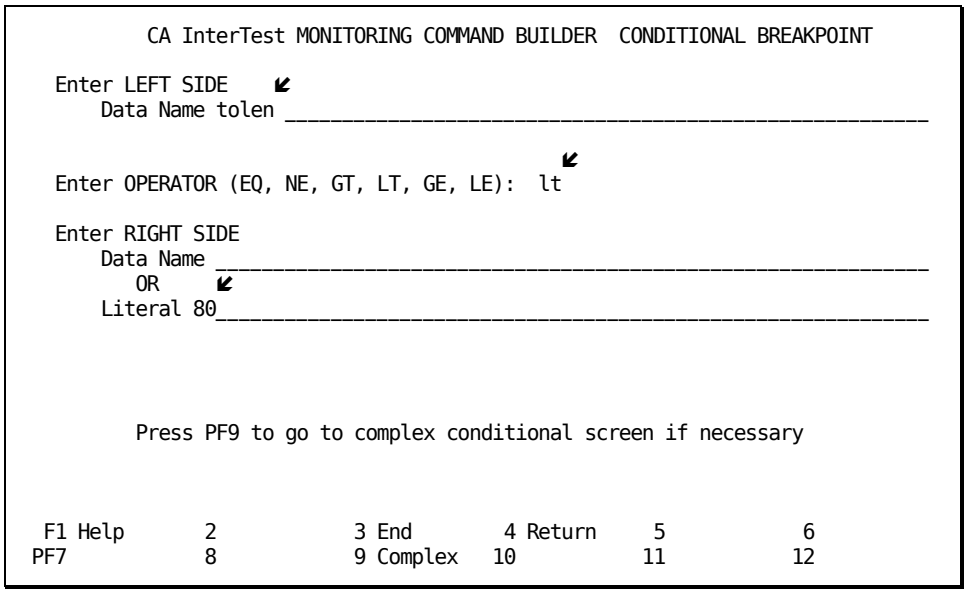
Valid Right Side Specifications for the Comparison			
ZERO ZEROS ZEROES	numeric value 0	HIGH-VALUE HIGH-VALUES	character value X' FF'
SPACE SPACES	blank (X'40')	LOW-VALUE LOW-VALUES	character value X'00'

Example

The following screen shows how to set a conditional breakpoint in a COBOL program:

- The left side of the comparison specifies the data name TOLEN.
- The relational operator is LT (less than).

- The right side of the comparison specifies the literal 80.



This breakpoint takes effect only if the value in the data name TOLEN is less than 80. The breakpoint locations are all the instructions, as previously specified on the Conditional Locations screen.

The Detailed Conditional Breakpoint Screen

Assembler and PL/I users who want to specify conditional breakpoints should complete the Conditional Locations screen and press Enter, the Conditional Breakpoint screen appears as follows. COBOL users access this screen by pressing **PF9** from the COBOL Conditional Breakpoint screen shown in the previous section.

```
          CA InterTest MONITORING COMMAND BUILDER   CONDITIONAL BREAKPOINT

❶  Enter LEFT SIDE of condition (select one):

     Data Name _____________________________________________________________
     Special keywords:   ______   Register # __         COBOL BLL # __


Enter OPERATOR (EQ, NE, GT, LT, GE, LE): __❷   Length:  Left  __❸  Right __❹

❺ Enter RIGHT SIDE of condition (select one):

     Data Name _____________________________________________________________
     Special keywords:   ______   Register # __         COBOL BLL # __
     Literal _______________________________________________________________

❻ Optional offset:  Enter +   @ OR % followed by a value.
     Left side  ____________________________   Right side ____________________

❼        Press PF9 to go to simple conditional screen



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

❶

Specifies the left side of the comparison that must be an area of core. Enter information in **one** of the following fields:

- Data name
- Register
- COBOL BLL cell
- Area identified by a CORE keyword

❷

Specifies the relational operator:

EQ for equal

NE for not equal

GT for greater than

LT for less than

GE for greater than or equal to

LE for less than or equal to

❸/❹

Optionally define the length of the left side or right side of the comparison. (See the Length of the Comparison section for details.)

❺

Specifies the right side of the comparison, which is an area of core or a literal. Enter information in one of these five fields: a data name, a register, a COBOL BLL cell, an area identified by a CORE keyword, or a literal. For more information, see the CORE Keywords section and the Literal Formats (see page 118) section later in this chapter.

❻

Optionally adjusts a CORE location by specifying offsets (displacements). Each offset must be preceded by one of the following operands:

- +
- -
- @–Indirect addressing below the 16-megabyte line
- %–Indirect addressing above the 16-megabyte line (XA systems only)

Note: Literals cannot be modified by offsets.

❼ For PL/I and Assembler version of this screen, the text line reads:

```
Use HELP or documentation for use of special keywords
```

Length of the Comparison

You can explicitly define how many bytes of either the left side or right side specification should be compared. Certain storage locations have implicit lengths:

- A register or COBOL BLL cell has an implicit length of four bytes.
- The storage locations referred to by the CORE keywords MXR, MXS, and TAL have implicit lengths of four bytes.

- The implicit length of a COBOL data name is its field length as defined in the DMAP.
- The length of a literal is the number of bytes it contains; any length specification is ignored.

Define both left side and right side lengths for a packed decimal (COMP-3) comparison. For all other comparisons, define only one length. If you define both lengths, the smaller length is used.

The maximum permissible length for the left side or right side is 16 bytes for packed decimal data, and 255 bytes for all other data types.

Literal Formats

For details on specifying literals, see Figurative Constants (see page 119) earlier in this chapter.

CORE Keywords

Specify the following CORE keywords on the left side or right side of the comparison:

Keyword	Function
CMAR	First byte of the EXEC CICS communications area for the task
CSA	First byte of the CSA
CURR	Next Assembler instruction to be executed
CWA	First byte of the CWA
CWK	First byte of the COBOL program's Working-storage
DSA	First byte of program's DSA
ITBE	First byte of the next instruction to be executed
LCL	First byte of the COBOL program's local-storage
MXR	CA InterTest for CICS maximum CICS request counter (implicit length = 4)
MXS	CA InterTest for CICS maximum storage counter (implicit length = 4)
OPFL	First byte of the Optional Features List
PGM=*	First byte of the monitored program.
PREV	Last Assembler instruction that was executed
R*nn*	A register (*nn* is a decimal from 1 to 15) (implicit length = 4)
TAL	CA InterTest for CICS tally fullword (implicit length = 4)
TERM=*	First byte of the terminal table entry of the current terminal

Keyword	Function
TGT	First byte of the COBOL TGT for the monitored task
TIOA	First byte of the first TIOA of the task
TWA	First byte of the TWA of the monitored task

Variable-Change Breakpoints

A variable-change breakpoint is a special type of conditional breakpoint that takes effect when the value of a specified COBOL or Assembler variable changes. If you set a variable-change breakpoint while the variable is also listed in the Keep window, the variable's new value immediately appears at the top of the breakpoint display when the breakpoint is triggered.

To set a variable-change breakpoint from a Source Listing screen, follow these steps:

1. Enter **v** in position 1 of any line in the program listing that defines or references the variable.
2. If the statement *references* the variable, place the *cursor* under any character in the variable name. (Omit this step if the statement defines the variable.)
3. Press Enter.

To set a variable-change breakpoint for a variable in the Keep window, follow these steps:

1. Enter **v** next to the variable name in the Keep window.
2. Press Enter.

Set variable-change breakpoints before or during program execution. If you set the breakpoint before execution, the variable's initial value is its value when it first becomes known to CA InterTest for CICS during execution. Any change from this initial value triggers the variable-change breakpoint.

Set Conditional Breakpoints from the Menus (see page 115) explains how to set variable-change breakpoints when the value of a variable increases or decreases.

Request Breakpoints

This section reviews setting request breakpoints from the source listing and from the menus.

Set Request Breakpoints from the Source Listing

Set request breakpoints to halt a program prior to CICS commands, macros, and other program calls (such as calls to DL/I, DB2, or SQL/DS). Instruct CA InterTest for CICS to halt the program before every CICS command, or a type of CICS command, such as File Control or Program Control commands. Specify that the program halts before specific commands, such as all READ or WRITE commands. Once the program is halted, use all of the CA InterTest for CICS facilities to inspect and modify main storage or auxiliary storage or to set additional options.

Set request breakpoints easily from the source listing by entering the RBP command. Once set, request breakpoints are identified by an R during program execution. Set them also from the Monitoring menus and they are most easily removed from the Monitoring Status display (STATUS command).

After completing the RBP menus, press **PF4 Return** until CA InterTest for CICS returns you to the Source Listing display.

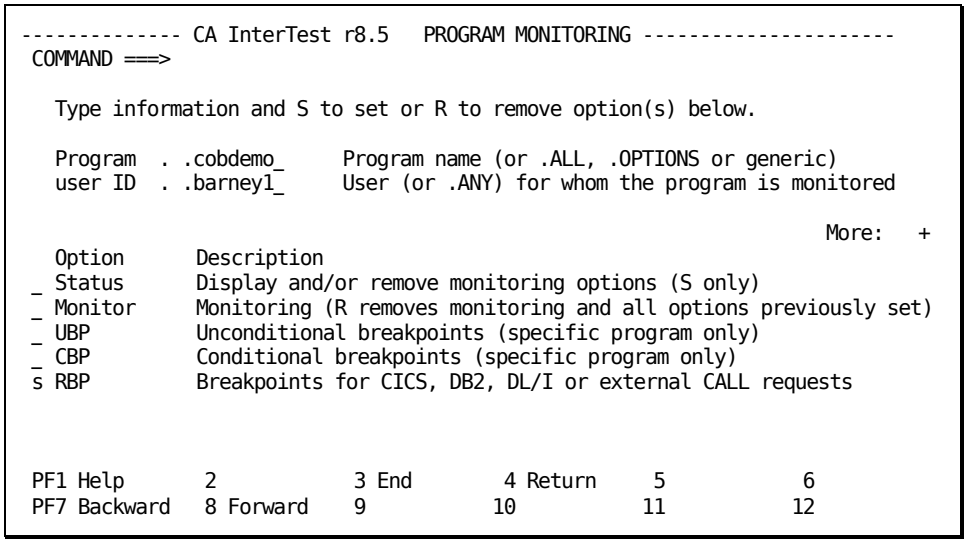
Set Request Breakpoints from the Menus

When viewing a program listing before or after a breakpoint, specify the RBP command to display the RBP menus.

Alternatively use the Program Monitoring (2.1), Transaction Monitoring (2.2), or Terminal Monitoring (2.33) menu to set request breakpoints. Complete the top of the menu according to how you are monitoring or want to monitor the program, terminal, or transaction, and select the RBP option on the bottom of the menu. Detailed steps on how to do this are given next for users unfamiliar with the monitoring menus.

1. Access the Primary Option Menu, and select **2 Monitor** to access the Monitoring Menu.
2. Select option **1 Program** to have the request breakpoints apply to a monitored program. Alternatively, select another monitoring option if you are monitoring by Transaction (2) or Terminal (2).
3. On the Monitoring submenu, complete the monitoring entry for the Program (Terminal or Transaction) and user ID, and then type **s** next to the RBP Option for Request Breakpoints and press Enter**.**

The following screen shows how to complete the Program Monitoring Menu (2.1) to set request breakpoints for the COBDEMO program being monitored for the user ID BARNEY1.



After specifying the RBP command from the Source Listing screen or completing the monitoring menu with the RBP option selected, CA InterTest for CICS displays the Request Breakpoint Selection menu.

```
      CA InterTest MONITORING COMMAND BUILDER  REQUEST BREAKPOINT SELECTION 13

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

❶ _ ALL commands                  ❷_ DL/I  ❸_ DB2   ❹_CALLS

     _ Address, Assign,          _ Storage Control    _ BMS
       Handles, Push, Pop        _ Program Control    _ Trace Control
     _ Terminal Control          _ Interval Control   _ Dump Control
❺    x  File Control              _ Task Control       _ Batch Data Interchange
     _ TD Control                _ Journal Control    _ BuiltIn Functions
     _ TS Control                _ Syncpoints         _ Sys Prog Functions

     Enter 'n' to stop only every n'th time:                          ____❻
     Term ID (or .ANY or .NO) where breakpoints will take effect:     ____❼
     Term ID (or .ANY) that will receive the breakpoints:             ____❽
     Statement no. of indirect command(s) to be executed:             ____❾
     user ID (or .ANY) who will execute the program:                  BARNEY1 ❿


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

On this screen enter an **x** next to the options you want to select:

❶

The program halts prior to all CICS commands.

❷

The program halts prior to all DL/I calls.

❸

The program halts prior to all calls to DSNHLI (for DB2 users) or prior to all calls to ARIPRDI (for SQL/DS users).

❹

The program is halted prior to calls to software that CA InterTest has been instructed to recognize at installation time. A second screen, where you specify the calls, displays.

❺

If you select a type of CICS command, CA InterTest for CICS displays a second screen. For example, if you select File Control, CA InterTest for CICS displays the File Control screen.

❻

If you do not want the program to halt before every occurrence of a command, specify in this field how often the program should be halted. For example, if you specify 3 the breakpoint occurs every third time the program reaches the command. This feature is especially useful when a CICS command occurs within a loop; for example, during a browse.

❼

Specify the ID of the terminal where the program must be executing for breakpoints to take effect. You can generally leave this field blank and have the default location apply.

- Enter .**ANY** to have breakpoints take effect at all terminals—even when the program executes without a terminal.
- Enter .**NO** to have breakpoints take effect only when the program executes without a terminal.

❽

Specify the ID of the terminal to receive breakpoint displays. You can generally leave this blank, and have the default location apply. Enter .**ANY** to have breakpoint screens displayed at the terminal where the program is.

❾

Specify the first indirect command statement number to take effect at this breakpoint.

❿

For secure CICS regions, specify the CICS user ID of the user who must be executing the program for the request breakpoints to take effect. If you do not have a secure CICS system, leave this blank.

Note: Your version of this screen may differ slightly depending on CA InterTest for CICS installation options.

When you have entered all of the necessary information, press Enter. If you entered an x next to a type of command, CA InterTest for CICS displays a second screen.

Example

If you selected File Control as the previous screen illustrates, CA InterTest for CICS displays the following screen.

```
    CA InterTest MONITORING COMMAND BUILDER   REQUEST BREAKPOINT SELECTION

             Set  one or more types of File Control commands/macros in:
                                 PROG=COBDEMO

               _ All commands            _ All macros

                 ➔  x READ
                 ➔  x WRITE
                    _ REWRITE
                    _ DELETE
                    _ UNLOCK
                    _ STARTBR
                    _ READNEXT
                    _ READPREV
                    _ ENDBR
                    _ RESETBR

               Enter 'n' to stop only every n'th time   ____


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

Using this screen specify that the program be halted at *all* File Control commands or macros, or select specific File Control commands. In this example, the user specified that program COBDEMO be halted at all READ and WRITE commands.

Remove Breakpoints

There are several ways to remove breakpoints. One of the easiest is to enter an **R** next to any breakpoint entry on the Monitoring Status display.This section discusses two other ways of removing breakpoints:

- From your Source Listing Breakpoint or Display
- From the Monitoring Menus (using STATUS command)

For more information:

Menus and Displays (see page 35)

Remove Breakpoints Flagged on Your Source Listing

You can remove breakpoints as quickly as you set them. Statements where you previously set unconditional or conditional breakpoints are flagged with a U or C**,** respectively, in column 1 of the line. During program execution, locations where CA InterTest for CICS stops because of a variable-change or request breakpoint are flagged with V or R, respectively.

- To change the type of breakpoint at a given location, first remove the existing breakpoint, and then set the next one.
- Remove automatic breakpoints only if the error is corrected. However, you can go around the error and start execution from another point.

The following table explains how to remove each type of breakpoint.

Type of Breakpoint	How to Remove
Unconditional, Conditional or Variable-change breakpoints	While viewing the source at a breakpoint, type an **x** over the U, C, or V; press Enter.
Unconditional or Conditional breakpoints set at all COBOL paragraph names, or Assembler or PL/I procedures	Either enter **x** to the left of the Procedure-Names or Symbol header line or Declare line in the Cross-Reference section of the listing and press Enter, or type an **R** next to the UBP or CBP entry on the Monitoring Status display and press Enter.
Unconditional or Conditional breakpoints set at all references to a COBOL or Assembler data name	Type an **x** over the C in the Cross-Reference section of the listing and press Enter, or type an **R** next to the UBP or CBP entry on the Monitoring Status display, and press Enter.
Request breakpoints	Specify the STATUS command then type **R** next to the RBP entry on the Monitoring Status display, and press Enter, or repeat the menu steps for setting the breakpoint, but enter an **R** next to the RBP option on the Monitoring menu. Complete the menus as you did when setting the request breakpoints.
***** All breakpoints	Remove monitoring for the program as discussed in the "Monitoring Menu Options" chapter. Do this from the Monitoring Status display**.**

Important! If the program is being monitored for .ANY user, remove the breakpoints instead of the monitoring entry. Removing monitoring for a program that others might also be testing is risky.

Note: If you set more than one type of breakpoint at the same location, CA InterTest for CICS flags and intercepts only one at a time—the first one you set. Once you remove the first one, the next one you set is flagged and activated, and so on.

For more information:

Breakpoint Activities (see page 135)
Resume and Execution Options (see page 213)

Remove Unconditional Breakpoints from the Menus

Remove unconditional breakpoints using the monitoring menus of the Primary Option Menu or using the STATUS command menu when at a source listing breakpoint display.

Note: An easy way to remove any breakpoint is to enter an **r** next to its entry on the Monitoring Status display (STATUS command).

To remove an unconditional breakpoint using the menus, access the Program Monitoring (2.1), Transaction Monitoring (2.2), or Terminal Monitoring (2.3) menu, and follow these steps:

1. Repeat the monitored entries for the following items:
   - Program, transaction, or terminal (whichever is being monitored)
   - User ID under which the entry is being monitored
2. Enter **r** next to the UBP Option for Unconditional Breakpoints and press Enter

The Remove Breakpoint Locations screen appears. COBOL, PL/I, and Assembler versions of the Breakpoint Locations screen are available. The screen is the COBOL version.

```
    CA InterTest MONITORING COMMAND BUILDER  COBOL BREAKPOINT LOCATIONS      11

REMOVE breakpoint locations for PROG=COBDEMO  in any of the following fields:
   (For qualification by nested program name, specify 'Programid:Label')
❶Para/Data _______________________________     _______________________________
     Names: _______________________________     _______________________________
            _______________________________     _______________________________
 Statement
❷Numbers: _____    _____    _____    _____    _____    _____    _____    _____

❸Offsets: _____    _____    _____    _____    _____    _____    _____    _____

           All Paragraph Names:  _  ❹                              After:_ ❺


          Term ID (or .ANY or .NO) where breakpoints will take effect:   ____❻
          Term ID (or .ANY) that will receive the breakpoints:           ____

 To remove by ID # shown in status display enter ID #'s: __ __ __ __ __ __ __❼


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

To remove all unconditional breakpoints from a program, enter **.ALL** in field ❶and press Enter. To remove one or more breakpoints, specify its ID number as shown on the Monitoring Status display in field ❸ and press Enter, or enter the information that you originally defined in fields ❶ through ❼ on the Set Breakpoint Locations screen and press Enter.

Remove Conditional Breakpoints from the Menus

To remove conditional breakpoints from the menus, use the monitoring menus of the Primary Option Menu.

Note: An easy way to remove any breakpoint is to enter an **r** next to its entry on the Monitoring Status display.

On the Program Monitoring (2.1), Transaction Monitoring (2.2), or Terminal Monitoring (2.3) menu, follow these steps:

1. Repeat the monitored entries for the:
   - Program, transaction, or terminal (whichever is being monitored)
   - User ID under which the entry is being monitored
2. Enter **r** next to the CBP Option for Conditional Breakpoints.
3. Press Enter.

The following Remove Conditional Locations screen appears.

```
 CA InterTest MONITORING COMMAND BUILDER  COBOL CONDITIONAL LOCATIONS     12

REMOVE breakpoint locations for PROG=COBDEMO  in any of the following fields:
  (For qualification by nested program name, specify 'Programid:Label')
❶Para/Data _____________________________     _____________________________
     Names:_____________________________     _____________________________
           _____________________________     _____________________________
  Statement
❷ Numbers: _____    _____    _____    _____    _____    _____    _____    _____

❸ Offsets: _____    _____    _____    _____    _____    _____    _____    _____

           All Paragraph Names:  _  ❹       All Instructions: _  After:_ ❺



         Term ID (or .ANY or .NO) where breakpoints will take effect:  ___ ❻
         Term ID (or .ANY) that will receive the breakpoints:          ____❼

         user ID (or .ANY) who will execute the program:           BARNEY1 ❽
  To remove by ID # shown in status display enter ID #'s  __ __ __ __ __ __ _❾


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

To remove all conditional breakpoints from a program, enter .ALL in field ❶ of this screen and press Enter. To remove one or more conditional breakpoints, enter the breakpoint ID numbers listed on the Monitoring Status display in field ❾and press Enter, or specify the information that you originally defined in fields ❶ through ❽ on the Set Conditional Breakpoints screen and press Enter.

Remove Request Breakpoints

To remove request breakpoints from the menus, use the monitoring menus of the Primary Option Menu.

Note: An easy way to remove any breakpoint is to enter an **r** next to its entry on the Monitoring Status display.

On the Program Monitoring (2.1), Transaction Monitoring (2.2), or Terminal Monitoring (2.3) menu, follow these steps:

1. Repeat the monitored entries for the:
   - Program, transaction, or terminal (whichever is being monitored)
   - User ID under which the entry is monitored

2. Enter **r** next to the RBP Option for Request Breakpoints.
3. Press Enter.

The Remove Request Breakpoints screen appears. Enter the information exactly as you originally defined it.

Chapter 5: Breakpoint Activities

This chapter discusses various breakpoint activities using CA InterTest for CICS facilities.

This section contains the following topics:

Examine the Source Listing Breakpoint Screen (see page 136)
Respond to Automatic Breakpoints (see page 143)
Work with Data at a Breakpoint (see page 146)
The Statement Trace Facility (see page 158)
Display or Modify Auxiliary Storage (FILE) (see page 161)
The Breakpoint Primary Option Menu (see page 163)
The CALLTRACE Facility (see page 164)
The CHANNEL Command (see page 165)
The Storage Facility (see page 166)
The Backtrace Facility (see page 167)
Additional Breakpoint Displays (see page 176)
The Detailed Breakpoint Screen (see page 178)
Indirect Commands (see page 187)
Edit Indirect Command Definitions (see page 197)
Indirect Command Syntax (see page 199)

Examine the Source Listing Breakpoint Screen

When CA InterTest for CICS monitors a program, it halts program execution each time it detects an error or a user-specified breakpoint; it then displays a breakpoint screen. The following screen shows an automatic breakpoint.

```
               CA InterTest  PROTDEM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt  4 Profile    5 Resume     6 Menu
     7 Backward  8 Forward   9           10 001 Verb  11 Backtrace 12 Status
--------------------------------------------------------------------------------
   _ 00880 CONTINUETASK.
     00881**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
❶ A   ==> ❷ADD +1 TO TASKNUM.
        ==>
 ❸    ==> ASRA abend (0C7) detected and prevented. Caused by invalid decimal
        ==> arithmetic data format.
        ==>
        ==>      Press PF1 for a detailed description.
        ==>
   _ 00883     IF TASKNUM = 1
   _ 00884         MOVE 'DMAPASR'  TO MAPNAME.
   _ 00885     IF TASKNUM = 2
   _ 00886         MOVE 'DMAPSUM'  TO MAPNAME.
   _ 00887     IF TASKNUM GREATER 2
   _ 00888         GO TO SENDENDMSG.
```

❶

Indicates the type of breakpoint that occurred:

A Automatic, detected by CA InterTest for CICS

U Unconditional

C Conditional

V Variable-change

R Request

❷

Indicates the statement or offset where the breakpoint occurred.

❸

These lines tell you why an automatic breakpoint occurred. For additional information, press **PF1**. To remove these lines, press **Enter** or scroll using **PF7** or **PF8** to see additional lines of source listing code.

Source Listing Breakpoint Commands

The command line at a Source Listing Breakpoint supports all of the commands described for a Source Listing Display plus the following commands: AUTOSTEP, BTRACE, CS, FO, GO, NEXT, RESUME, RESETBKP, and RUN. These commands are defined in the following table. In addition, use the command line to enter CA InterTest for CICS single-line commands for CNTL, CORE, and FILE. See the online help and other chapters in this guide for information on single-line commands.

Note: You cannot use this command line to enter CA InterTest for CICS system-wide CNTL options.

Command	PF Key	Description
ABEND	None	Display the Abend Breakpointed Task screen
ABI ON	None	Turn abend intercept on (default). This intercepts all CICS abends and produces an automatic breakpoint screen providing a CICS HANDLE ABEND LABEL/PROGRAM is not active. If you want to intercept CICS abends for programs having an active CICS HANDLE LABEL/PROGRAM, see the ABI FORCE command.
ABI OFF	None	Turn abend intercept off. This causes CA InterTest for CICS to stop intercepting CICS abends. No automatic breakpoint screen appears and the program abends.
ABI FORCE	None	Turn Abend Intercept on for all CICS ABENDS, even if a program has an outstanding HANDLE ABEND active or issues a CICS HANDLE ABEND command. Use this option when you want an automatic breakpoint screen for programs that also have active HANDLE ABENDS to process their own errors.
ADVANCE n ADV n A n	None	Advances n entries in the statement trace table and displays the resulting statement. (If n is not specified, it defaults to 1.) If data monitoring is active, past data values for the statement are also displayed. Use this command to redisplay the execution sequence of a program. Statement tracing must be active for the ADVANCE command to function (see the TRACE option). **Note:** This command is only valid for COBOL programs.
AUTOSTEP x y AS x y	None	Resume execution x verb(s) at a time displaying the current source statement for a y interval of time. This command is available only at a breakpoint. Default: One verb at at time until next CALL/COMMAND.

Command	PF Key	Description
BOTTOM BOT	None	GO to bottom of source.
BPO	None	Jump to Breakpoint Option screen.
BTRACE	PF11	Backtrace. Displays the backtrace summary. This command is available only at a breakpoint.
BWD	PF7	Scroll backward. Set the scroll amount on the Source Listing Profile. Does not apply within the Keep window.
CALLTRACE	None	Shows the program call flow leading up to the currently breakpointed program. Select programs having symbolic support from the list and display their data areas. Select channels at a specific program level from the CALLTRACE program list.
CHANNEL	None	Displays the currently allocated channel and the containers in the channel. Display the data within a container from the selection list .This feature is only valid in CICS releases that support CHANNELs and Containers.
CNTL	None	Displays the CNTL Command Menu. **Note:** The Monitoring Menu (=1.2) is an alternative to using the CNTL menu.
COMPOSITE	None	Set composite monitoring options for monitoring of sub-programs.
CORE	None	Displays the Main Storage menu. Select an option to view a main storage (CORE) display.
COUNTS SHOW	None	Turns on the code coverage COUNTER display. To activate code coverage, see Customizing the Source Listing Profile or issue PROFILE command.
COUNTS NOSHOW	None	Turns off the code coverage COUNTER display. To deactivate code coverage, see Customizing the Source Listing Profile or issue the PROFILE command.
COUNTS RESET	None	Resets the program code coverage statement counters to zero.
COUNTS op value	None	Turns on the code coverage COUNTER display and highlights program statements that match the criteria. The source listing is repositioned to the first statement matching the criteria. Op values: EQ or =, NE or <>, LT or <, GT or >, LE or <= or =<, GE or >= or => Values: 0 to 9999

Command	PF Key	Description
COUNTS NEXT	None	Repositions the source listing to the next statement that matches the previously specified COUNT op value criteria.
COUNTS ALL	None	Removes previously specified COUNT op value criteria and redisplays source listing without special COUNTER highlighting.
CS	None	Go to current breakpoint source statement. This command is available only at a breakpoint.
DATAMON DM	None	Displays the Monitoring Command Builder menu to set and remove statement tracing and data monitoring. **Note:** This command is only valid for COBOL programs.
DATAMON ON DM ON	None	Turns data monitoring on. This causes CA InterTest for CICS to capture data values for each executed statement in the program. You are then able to redisplay these values at a breakpoint, by tracing backward and forward through the entries in the statement trace table using the PREV and ADVANCE commands. DATAMON ON forces TRACE ON. **Note:** This command is only valid for COBOL programs.
DATAMON OFF DM OFF	None	Turns data monitoring off. This causes CA InterTest for CICS to stop capturing data values for each executed statement in the program. In addition, you can no longer display captured data values when tracing backward and forward through the entries in the statement trace table using the PREV and ADVANCE commands. **Note:** This command is only valid for COBOL programs.
DISPLAY data-item D data-item	None	Displays the storage of a data name.
DOWN	None	Shift listing down.
END	PF3	Returns to the prior display or menu.
FILE	None	Displays the Auxiliary Storage menu. Select an option to view a file or queue.
FIND F	None	Find a string
FO	None	Find hexadecimal offset. This command is available only at a breakpoint.
FP	None	Find paragraph.
FS	None	Find statement number.

Command	PF Key	Description
FWD	PF8	Scrolls forward the amount indicated on the source listing Profile. Does not apply within the Keep window.
GO	None	Continue until the next intercept occurs. This command is available only at a breakpoint.
HELP	PF1	Displays help for the source listing facility.
IC	None	Jump to Indirect Command screen.
ITST	None	Displays the Primary Option menu.
KEEP data-item K data-item	None	Add a data-item to the static keep window
LEFT	None	Shift listing left.
LOCATE LOC L	None	Locate line number or part of the listing.
MARGIN MAR	None	Shift listing to specific margin.
MENU	PF6	Displays the Breakpoint Primary Option menu.
MONITOR	None	Sets monitoring for the listed program.
NEXT N	PF10	Execute the next verb in a program. This command is available only at a breakpoint.
NEXT OVER	None	Continue the test session when stopped on a PERFORM or CALL until the statement following the PERFORM or CALL is encountered, where a break point occurs.
NEXT RETURN	None	Continue the test session when stopped within a paragraph or subroutine until the session encounters the statement following the PERFORM or CALL that invoked the current paragraph or subroutine, where a break point occurs. TRACE ON must be enabled.
OFFALL	None	Removes all breakpoints set by active USER or TERMINAL. Active USERID is used if a valid USERID is signed on. Otherwise, the active TERMINAL is used. ALL breakpoints for ALL programs/transactions/ are removed that were set by the active USERID or active TERMINAL.
OFFALL TERM	None	Removes all breakpoints set by active TERMINAL.
OFFALL USER	None	Removes all breakpoints set by active USER.

Command	PF Key	Description
PO	None	Set or remove storage protection options
PREV n P n	None	Backs up n entries in the statement trace table and displays the resulting statement. (If n is not specified, it defaults to 1.) If data monitoring is active, display also past data values for the statement. Use this command to step backward through the execution of a program. Statement tracing must be active for the PREV command to function (see the TRACE option). **Note:** This command is only valid for COBOL programs.
PROFILE	PF4	Displays the source listing Profile, where you change settings for the current session.
RBP	None	Set or remove Request Breakpoints
RESETBKP	Clear	Repositions breakpointed task at current breakpoint.
RESUME	PF5	Resumes breakpointed task at next sequential instruction. This command is available only at a breakpoint.
RIGHT	None	Shift listing right.
RO	None	Set or remove replacement options
RUN	None	Resume execution ignoring all breakpoints. This command is available only at a breakpoint.
SET	None	Initialize a data-item's value
SO	None	Set or remove special options
SDFDATA	None	Displays data in Structure Display Format for the current program.
SDFHEX	None	Displays data in hexadecimal/character format for the current program.
STATUS	PF12	Displays the Monitoring Status report (Option 2.4 on the Primary Option Menu) for the current program. Remove monitoring, breakpoints, and options directly from the Status display.
STATUS ALL	None	Displays the Monitoring Status report for all programs, transactions, and terminals CA InterTest for CICS is monitoring in the region.
Storage	None	Displays a listing off all USER and CICS class storage associated with the currently breakpointed task (most storage acquired by CA InterTest for CICS is not displayed), this command also checks the crumple zones of each piece of storage for validity.

Command	PF Key	Description
TRACE TR	None	Displays the Monitoring Command Builder menu to set and remove statement tracing and data monitoring. **Note:** This command is only valid for COBOL programs and is unrelated to the Backtrace facility.
TRACE ON TR ON	None	Turns statement tracing on. The number of entries to be saved in the statement trace table is determined by the STMTTRCE option in IN25OPTS. The default is 1000. **Note:** This command is only valid for COBOL programs and is unrelated to the Backtrace facility.
TRACE OFF TR OFF	None	Turns statement tracing off. TRACE OFF forces DATAMON OFF. **Note:** This command is only valid for COBOL programs and is unrelated to the Backtrace facility.
TOP	None	Go to top of source.
UP	None	Shift listing up.
=x.y.z =**1**.x.y.z	None	Fast-path to **Breakpoint** Primary Menu Option x.y.z, discussed in the "Breakpoint Activities" chapter. Equivalent ITST Primary Option Menu fast-path entries are =**1**.x.y.z.
<{nnnn}	None	Used in the Keep window to scroll data item storage left nnnn bytes.
>{nnnn}	None	Used in the Keep window to scroll data item storage right nnnn bytes.

For more information:

Monitoring Menu Options (see page 227)

Source Listing Breakpoint PF Keys

PF Key	Function
PF1 Help	Displays a Help menu relating to the current Source Listing Breakpoint screen.
PF2	Unassigned.
PF3 Det Bkpt	Displays the Breakpoint Information screen for the current breakpoint. This function is available only at a breakpoint.

PF Key	Function
PF4 Profile	Displays the Source Listing Profile screen.
PF5 Resume	Continues program execution at the next sequential instruction. This function is available only at a breakpoint.
PF6 Menu	Displays the Breakpoint Primary Options Menu discussed in this chapter.
PF7 Backward	Scrolls the compiler output backward one page or the amount set on the source listing profile. Does not apply within the Keep window.
PF8 Forward	Scrolls the compiler output forward one page or the amount set on the source listing profile. Does not apply within the Keep window.
PF9	Unassigned
PF10 001 Verb	001 Verb/Instr/Stmt or Autostep single steps or auto steps through execution, as indicated by the PF key label. Use PF4 to specify the stepping function and amount on the source listing Profile. This function is available only at a breakpoint.
PF11 Backtrace	Displays the Backtrace Summary. This function is available only at a breakpoint.
PF12 Status	Displays the Monitoring Status for the current program. Remove breakpoints and monitoring directly from the Status report.
PF19	Scrolls forward within the Keep window.
PF20	Scrolls backward within the Keep window.
Clear	Redisplays the current breakpoint.

Respond to Automatic Breakpoints

An automatic breakpoint occurs when CA InterTest for CICS detects an error. When a program is stopped at an automatic breakpoint, either correct the error or go around it.

To find out what caused the error and for instructions on how to fix it, press **PF1** from any automatic breakpoint display with the error message displayed. CA InterTest for CICS Help provides additional information about the specific error. In most cases, Help recommends how to handle the error.

Example

The following screen explains how to correct an AEIL abend:

```
                  CA InterTest  INTERACTIVE HELP FACILITY
 TUTORIAL: ERROR MESSAGE AEIL

The dataset name referred to in the DATASET option cannot be found.
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.
                                                                    (continued)

--------------------------------------------------------------------------------

  ENTER N FOR NEXT PAGE, P FOR PRECEDING PAGE, F FOR FIRST PAGE, OR
  M FOR RETURN TO PREVIOUS MENU. PRESS CLEAR TO EXIT.          SELECTION ==> N
```

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**.

Bypass the Monitoring of a Statement that Triggered an Automatic Breakpoint

Sometimes CA InterTest for CICS stops you at an automatic breakpoint when the program is working according to your specifications. These are usually instances when you are intentionally violating a CICS coding standard. To continue program execution without removing the error, bypass the statement, as described next, or set a CA InterTest for CICS monitoring option so that you are not stopped again (such as NOM). NOM is supported in Release 5.1 and higher.

Important! Use this function with care! It bypasses monitoring only of the individual Assembler instruction. For high-level COBOL or PL/I programs, there might be more than one Assembler instruction per statement, which could trigger more than one automatic breakpoint for a single statement. As a result, you might have to invoke this function more than once to continue past a statement stopped at an automatic breakpoint.

There are two methods of bypassing an automatic breakpoint: temporarily and permanently. A temporary override works for the current breakpoint only. A permanent bypass works each time the statement is executed.

For more information:

Monitoring Menu Options (see page 227)
Special Monitoring Situations (see page 267)

Temporary Override

To temporarily bypass the monitoring of a statement that triggered an automatic breakpoint:

1. Press **PF6** to display the Breakpoint Primary Option Menu.
2. Select Option 5 Override Automatic Breakpoint.

Note: For fast-path entry, enter **=5** on the command line of the Breakpoint display.

The override lets the statement that caused the breakpoint to be executed for this instance, only.

Permanent Bypass

To permanently bypass the monitoring of a statement that triggered an automatic breakpoint:

1. Overtype the A on the Source Listing Breakpoint with **b**.
2. Press **PF5** or **PF10** to resume execution. This lets the statement causing the breakpoint to be executed; that is, the bypass function lets the error to occur.

Important! This command may be password protected.

Work with Data at a Breakpoint

When CA InterTest for CICS detects a breakpoint, it halts the program before the instruction at the breakpoint location is executed. While at a breakpoint, CA InterTest for CICS preserves the task's main storage for examination or modification prior to task resumption. Here is a summary of how to work with your data at a breakpoint. Each function is explained in this chapter.

- **View or Search for Data**

  View any portion of your compiler output and search for occurrences of data strings.

- **Keep Data in Window**

  Keep the values of data items, displayed up to 6 at a time, at the top of your source listing in the Keep window. This lets you observe changes in their values as the program executes without having to leave the source listing display.

  When the Keep window is active, the header lines listing the CA InterTest for CICS options and PF key functions are replaced with a command line. To view options and PF keys, press **PF4**, then press Enter to return.

- **Code Coverage Counters in Window**

  Display a counter for each line of code and check its verb execution count. This lets you observe changes in the execution value as the program executes without having to leave the source listing display.

- **Display or Modify Main Storage**

  Display and dynamically change data in main storage at any breakpoint. If there is an error, fix it in main storage, and then continue execution as if the error never existed.

  Online access to main storage means you do not have to wait for and analyze a dump. To make accessing main storage easy, CA InterTest for CICS provides a structured display of main storage with data names on the left and both hexadecimal and character values on the right.

- **Display or Modify Auxiliary Storage**

  View, modify, or create test data using the FILE transaction. At a breakpoint, verify test data, and even create test records.

For more information:

Accessing Auxiliary Storage: FILE (see page 441)

Search for Data

Ask CA InterTest for CICS to define, search for, and display any character string (such as a data item, label, or paragraph name). This is a quick way to go to different areas in a listing when you want to set a breakpoint or check your code.

Search for data using the Search=field, or by entering a FIND or LOCATE command on the command line. If you know the line number, hex number, or the label that you want to view, use the line commands to position the listing accordingly. For details, 8 (see page 148) later in this chapter.

Find a Definition or Search Using the Search = Field

Use the Search= field to define or search for data as follows.

To display the definition of a data item, follow these steps:

1. Enter a data item name up to 31 characters in the Search= field.
2. Press Enter to begin the search. CA InterTest for CICS highlights the data item and its definition.

To search for a data item, follow these steps:

1. Enter up to 31 characters in the Search= field.
2. Specify the search direction in the Option # field. (Option # 9 is Search Forward; Option # 10 is Search Backward.)
3. Press Enter to begin the search.

For COBOL nested programs, CA InterTest for CICS first searches the current nested program (indicated in the NESTED= field) for the specified data item. If the data item is not found in the indicated program, CA InterTest for CICS then searches the main program and other nested programs for the item.

Note: It is not necessary to enter the entire data item, label, or paragraph name; enter the first few characters to begin your search.

Example

If you wanted to search for TASKNUM, you could enter TASK, TASKN, and so on, in the Search= field.

The Online Listing

Use the command line commands to position or shift the view of the listing. Go to the top or bottom of the listing, shift the listing to the right or left a specific number of characters, position the listing to a specfic line.Described next are two additional commands that are available only at a source listing breakpoint. These commands enable you to position a listing at specific hex number, or position the listing to a specific label or to the current breakpointed line.

For more information:

Source Listing Facility (see page 47)

Position Listing to a Hex-Offset

To position the listing to a specific hex-number, enter:

F0 *hex-number*

Hex-number

Indicates any hex number within an Assembler program.

Note: FO is available only during breakpoint processing.

Position Listing to Current Breakpoint Processing

To position the listing to the current breakpointed line, enter:

CS

Note: CS is available only during breakpoint processing.

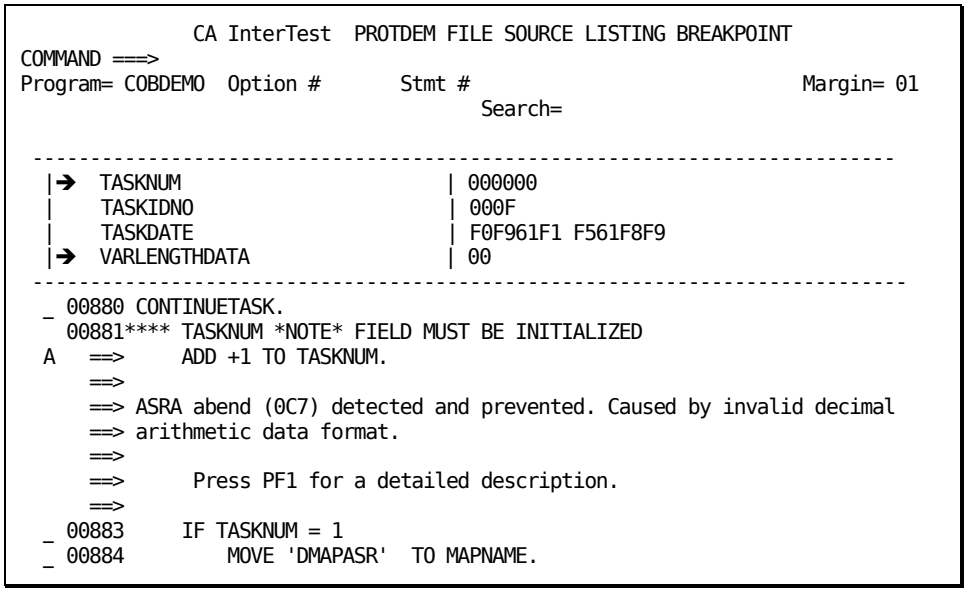
The Keep Window

Display items from main storage in a window at the top of the Source Listing display. Keeping items in the Keep window lets you easily compare values and observe changes in values. It also lets you to dynamically modify the displayed values as the program executes.

Add a Storage Item

Display up to six data items at one time in the Keep window whenever the source listing is displayed. Identify the data items whose values you want to observe. If you identify more than six items, the first six items that you specify are displayed. When one item is removed from the Keep window, another item appears. If there are more than six items in total, view these by scrolling forward.

The following screen shows four data items displayed, in hexadecimal format, in the Keep window on the source listing breakpoint display.



To add a data item from where it is **defined,** follow these steps:

1. Enter **k** to the left of the statement defining it.
2. Press **Enter**. CA InterTest for CICS places the data item in the Keep window.

To add a data item from where it is **referenced,** follow these steps:

1. Enter **k** to the left of the statement referencing it.
2. Place the **cursor** under any character in the data item.
3. Press **Enter**. CA InterTest for CICS places the data item in the Keep window.

Note: Data items remain in the Keep window until you remove them or until you turn off CA InterTest for CICS monitoring for the program—except for items added using the AUTOKEEP facility. AutoKeep items appear in high intensity in the Keep window; permanent Keep items appear in low intensity.

Using AUTOKEEP

AUTOKEEP lets you to display items in the Keep window concerning the currently highlighted line, in addition to any data items you requested displayed permanently.

AUTOKKEEP is active when the IN25OPTS parameter SLBAKEEP is set to Yes, which is the default. However, use the Source Listing Profile Screen to turn AUTOKEEP off or on again at any time.

Change an AutoKeep Item into a Permanent Keep Item

To change an AutoKeep item into a permanent item, follow these steps:

1. Enter **k** to the left of the data item.
2. Press **Enter**. The item now appears in low intensity indicating that it is a permanent item.

Scroll Forward and Backward

Scroll forward and backward through the Keep window when there are more than six Keep Table entries. Up to six entries appear in the Keep window at one time. If there are more than six entries, the heading line, MORE: + (plus) displays in the Keep window indicating that there are more entries to view forward. The additional entries appear in the next window as you scroll forward. The heading line, More: - (minus), indicates that there are more entries to view backward. Use **PF19** to page forward to the next Keep window; use **PF20** to page backward.

Scroll Left and Right

Scroll each data item in the Keep window left or right by using the arrow (< >) line commands. To scroll a data item left, type **<** in the first position of the line command area and press Enter. A **<** line command alone scrolls the data left the width of the display area. To scroll a specific number of bytes specify a numeric offset value after the line command, such as **<5**. This causes the data to scroll 5 bytes left or +5 bytes from the start of the data. To scroll back to the start of the data enter **>5** or **>** without an offset. Data item values with lengths that are less then the display area cannot be scrolled. A data item value cannot be scrolled right, **>** unless it has been first scrolled left (**<**).

Remove Items

To remove a data item from the Keep window, follow these steps:

1. Enter **x** to the left of the data item.
2. Press **Enter**.

When all data items are removed, the command line and the Keep window are replaced by the options and PF key functions.

View and Change Main Storage Display from the Keep Window

When the Keep window is active, CA InterTest for CICS replaces the title lines for the CA InterTest for CICS options and PF key functions with expanded source display area. This is the No Titles format. The No Titles format lets you to use CA InterTest for CICS commands and the options and PF keys to perform CA InterTest for CICS functions.

- To see the options and PF key functions, enter **PROFILE** in the command line and press Enter
- To exit the Profile, press Enter

For each item in the Keep window, CA InterTest for CICS displays the data name and its value in main storage in hexadecimal or character format depending on the SDF profile setting. Use the PF2 key to toggle between hexadecimal and character formats.

To request a main storage display, follow these steps:

1. Enter **d** in column 1 to the left of the item listed in the Keep window.
2. Press **Enter** to process your request. CA InterTest for CICS displays the main storage for the data item and all items below it in the same structure.

To modify data values using a pre-formatted MOVE command, follow these steps:

1. Enter **m** in column 1 to the left of the item listed in the Keep window.
2. Press Enter to make the change. CA InterTest for CICS displays the MOVE command screen. For details, Display or Modify Main Storage (CORE) (see page 152) later in this chapter.

To change the values of the data items, follow these steps:

1. Overtype the bytes displayed in the Keep window in either hexadecimal or character format.
2. Press Enter to change the values that appear in the Keep window.

An effective debugging tool is to observe data values in the Keep window as you single-step through execution using **PF10**. You may also have CA InterTest for CICS automatically repeat single-stepping every few seconds. Use the Profile display to set the PF10 value for automatic single-stepping.

For more information:

Source Listing Facility (see page 47)

Display or Modify Main Storage (CORE)

Display and modify easily the value of any data item or register in main storage directly from the source listing display screen. All main storage functions are functions of the CORE facility, but you do not have to access the CORE menus to use the functions described here.

Note: When you request main storage displays directly from your breakpoint screen, CA InterTest for CICS responds with a structured display whenever the request is for a data item defined in a structure.

Main storage functions include the following benefits:

- Displaying the value of a data item
- Modifying the value of a data item with a formatted MOVE command
- Modifying a data item by overtyping the main storage display or Keep window display
- Displaying and modifying program registers

The first three functions are discussed in the sections that follow; displaying and modifying program registers are discussed in the chapter "Source Listing Facility."

Note: Keep storage items displayed directly on your Source Listing Breakpoint screen by using the Keep window. For details, The Keep Window (see page 148) earlier in this chapter.

Display the Value of a Data Item

Display data items from where they are defined (for example, from the Working-Storage, Local-Storage, or Linkage sections), or from where the data items are referenced in the code.

To display the value of a data item from where it is defined, follow these steps:

1. Enter **d** to the left of the line defining it. For multiple data items, enter **d** next to the definition of each item that you want to display.
2. Press **Enter**. CA InterTest for CICS responds with a main storage display.

If you requested more than one item for display, CA InterTest for CICS responds with a structured CORE storage display for the first item selected. The message line on the screen prompts you to press **Clear** to view the next item, or press **PF3** to cancel the remaining requests. After the last item is displayed, press **Clear** to return to the Source Listing Breakpoint display.

To display the value of a data item from where it is used in the code, follow these steps:

1. Enter **d** to the left of any line in the compiled listing that contains the item.
2. Move the cursor under any alphanumeric character in the data item.
3. Press **Enter**.

CA InterTest for CICS responds with a main storage display. The first field on the display screen is the value of the data item you requested.

Important! If an Assembler variable is redefined in the code with a using statement other than the original DSECT in which it was originally contained, then the display of the data item should be made from where it is defined, not from where referenced.

Use the Cursor

Follow these rules when using the cursor to display main storage:

- Place the cursor under any character in a data name, Assembler register, or Assembler label.
- Do not place the cursor under a comma, parenthesis, COBOL verb, Assembler masks, Assembler data items in the form A+B or A-B+n, or under any comment area.

To display complex data items, use one of the following methods:

- Display qualified data names as follows: For X of Y, place the cursor under X in any line of code.
- Display subscripted or indexed data names as follows:

  For Y(s1, s2, s3):

  - To get the value of Y at s1, s2, s3, place the cursor under Y
  - To get the value of s1, place the cursor under s1

Example

To display the value of TASKNUM from the Automatic Breakpoint screen shown next, overtype the A with d, move the cursor under any letter in TASKNUM, and press Enter.

```
00880 CONTINUE TASK.
     00881**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
    d  ==> ADD +1 TO TASKNUM.
       ==>
       ==> ASRA abend (0C7) detected and prevented. Caused by invalid decimal
       ==> arithmetic data format.
       ==>
```

CA InterTest for CICS produces the following display of main storage for TASKNUM:

```
   CA InterTest   MAIN STORAGE UTILITY
  Starting at Address =  2142A8               Hexadecimal            Character

 02 TASKNUM                          | 000000                     | ...
 02 TASK TEXT                        | 00000000 00000000 00000000 | ............
                                     | 00000000 00000000 00000000 | ............
                                     | 00000000 00000000          | ........









  ______________________________________________________________________________
_
PFKEYS 1 Help     2           3 End       4 Return    5            6 Menu
       7 Backward 8 Forward   9 Caps Off 10           11 Redisplay 12 Structure
 CORE='TASKNUM'
 CAIN0452 FIELD DOES NOT CONTAIN A VALID PACKED DECIMAL (COMP 3) VALUE
```

Important! If you enter d to the left of a statement and forget to place the cursor under the data item, you will see the object code for the statement instead. Press **Clear** or **PF3** to return to the Source Listing Breakpoint.

Modify a Data Item by Overtyping the Main Storage Display

From any CA InterTest for CICS main storage display or Keep window, modify main storage by following these steps:

1. Overtype the desired bytes in the character or hexadecimal area.
2. Press **Enter** to process the change.

Use this method to correct invalid data or data that was not initialized. Remember that these modifications are **dynamic—**they are one-time fixes that do not prevent the same error from occurring again.

Example

In the previous screen, you could initialize TASKNUM by overtyping the final 0 in the hexadecimal field for TASKNUM with **C** and pressing Enter.

Notes:

- Display and modify task-related and system-related areas of CICS main storage using the CORE facility menus or commands.
- If you try to change a password-protected storage area not owned by your program, CA InterTest for CICS prompts you to enter the password.

For more information:

Accessing Main Storage: CORE (see page 375)

Modify the Value of a Data Item with a MOVE Command

COBOL and PL/I

CA InterTest for CICS provides a preformatted MOVE command that lets you easily modify the value of a COBOL or PL/I data item. Request the MOVE command from the following places:

- Where the data item is defined in the source listing
- A Keep window
- Any place in the listing where the data item is referenced

To modify a data item from the statement where it is defined or from a Keep window, follow these steps:

1. Enter **m** to the left of the statement. To modify more than one data item, enter **m** next to each item in the order in which they appear.
2. Press **Enter**. CA InterTest for CICS generates a fill-in-the-blank MOVE command for each data item selected.

To modify a data item from where it is referenced, follow these steps:

1. Enter **m** to the left of any statement in the listing containing the data item.
2. Place the cursor under the data item.

3. Press Enter. CA InterTest for CICS generates a fill-in-the-blank MOVE command for each data item selected.

   Note: This function is not supported for Assembler users.

Example

If you enter **m** to the left of the fields TASKNUM and TASK-TEXT from their definitions in Working-storage and press Enter, CA InterTest for CICS displays the following formatted MOVE commands.

```
              CA InterTest  PROTDEM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt   4 Profile    5 Monitor    6 Menu
     7 Backward  8 Forward   9           10           11           12 Status


MOVE ____________________________________________________________________

 to

     TASKNUM

MOVE ____________________________________________________________________

 to

     TASKTEXT

Overtype Underscores with a DataName, Figurative Constant,
Alphanumeric Literal, or Numeric Literal
```

To change the value of a data item, follow these steps:

1. Enter one of the following in the entry field:
   - A data name
   - ZEROS, SPACES, LOW-VALUES, HIGH-VALUES, QUOTES
   - A numeric literal with or without a leading + or – sign
   - An alphanumeric literal enclosed in single quotes
   - ALL *literal* to fill the data item with a specified literal string. For example: ALL '–' fills the field with dashes.

2. Press Enter to execute the MOVE command. CA InterTest for CICS responds with a main storage display showing the result of the first MOVE. For multiple moves, the storage displays are shown one at a time; press **Clear** to process the next one.

   Note: If you do not press **Clear**, the remaining MOVE commands will not execute.

3. From the CORE display, press **Clear** or **PF3** to return to the source listing breakpoint display.

Note: All modifications to main storage are dynamic. This means that the changes are **one-time fixes** to current values, and will not prevent the same error from occurring again. For a more permanent correction, specify a set of indirect commands to be executed at a particular program location. For details, Indirect Commands (see page 187) later in this chapter.

The Statement Trace Facility

The Statement Trace Facility lets you view the logic flow of your program at the statement level. When used in conjunction with data monitoring, it also lets you to view data values from 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. If a statement trace table was not created for the transaction, it will be created prior to executing the next statement in the program. The number of entries in the table is defined by the STMTTRCE option in IN25OPTS. Once statement tracing is activated, CA InterTest for CICS saves information about each statement in the program, as it is executed. When all of the statement table entries are used, the table will *wrap around* and the oldest entry is reused for the current statement.

Use the DATAMON command, or DM, to enable data monitoring for a program. Since the statement trace facility is required for data monitoring, it will also be enabled if it is not already active. Once data monitoring is active, CA InterTest for CICS captures the data values for each statement in the program, as it is executed.

Navigate the Statement Trace Table

Once statement information is saved in the trace table, use the PREV and ADVANCE commands to determine the execution path of the program, by stepping backward and forward through the executed statements.

Use the PREV, or P, command to cause the display to back up one or more entries in the statement trace table. Consider the following breakpoint:

```
    CA InterTest for CICS r8.5  - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ===> prev
Program= COB2DEMO Option #       Stmt #                              Margin= 01
                                        Search=
--------------------------------------------------------------------------------
 _____ TASKNUM                         | 000000
---------+----------------------------------------------------------------------
 _ 000531 CONTINUE-TASK.
   000532**** 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.
      ==>
 _ 000534     IF TASKNUM = 1
 _ 000535         MOVE 'DMAPASR'  TO MAPNAME.
 _ 000536     IF TASKNUM = 2
 _ 000537         MOVE 'DMAPSUM'  TO MAPNAME.
 _ 000538     IF TASKNUM GREATER 2
 _ 000539         GO TO SEND-END-MSG.
 _ 000540     GO TO REWRITE-TSQ.
```

The PREV command will back up to the previous entry in the table, resulting in the following display:

```
CA InterTest for CICS r8.5   - PROTSYM FILE SOURCE LISTING BREAKPOINT
COMMAND ===> prev
Program= COB2DEMO Option #       Stmt #                              Margin= 01
                                        Search=
--------------------------------------------------------------------------------
 _____ TASKNUM                         | 000000
---------+----------------------------------------------------------------------
000522                MOVE '   00209 '        TO DFHEIV0
_ 000523       CALL 'DFHEI1' USING DFHEIV0.
_ 000524       IF EIBAID = DFHENTER GO TO CONTINUE-TASK.
_ 000525       IF EIBAID = DFHCLEAR GO TO SEND-END-MSG.
_ 000526       IF EIBAID = DFHPF3 GO TO SEND-END-MSG.
_ 000527       IF EIBAID = DFHPF15 GO TO SEND-END-MSG.
_ 000528       IF EIBAID = DFHPF2 GO TO EXPANDED-DEMO.
_ 000529       IF EIBAID = DFHPF14 GO TO EXPANDED-DEMO.
_ 000530       GO TO SEND-FIRST-SCREEN.
_ 000531 CONTINUE-TASK.
   000532**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
 A    ==>     ADD +1 TO TASKNUM.
 _ 000534     IF TASKNUM = 1
 _ 000535         MOVE 'DMAPASR'  TO MAPNAME.
 _ 000536     IF TASKNUM = 2
 _ 000537         MOVE 'DMAPSUM'  TO MAPNAME.
```

Statement number 524 was the last statement executed prior to the abend at statement number 533. To back up 10 entries in the table, specify PREV 10. After using the PREV command to back up one or more entries in the statement trace table, use the ADVANCE, or A, command step forward through the table. Return at any time to the current statement using the CS command or the clear key.

View Past Data Values

While navigating the trace table, display the data values that were captured as each statement was executed, if data monitoring is active. If the AUTOKEEP display is ON, the past data values associated with an executed statement automatically appear in the Keep window when the statement is displayed. In addition, use the DISPLAY, KEEP, and CORE commands to display past data values for any data item or COBOL language area (for example, TGT). However, these values are protected and, therefore, cannot be updated while viewing past data values.

In general, past data values are not captured for database areas or CICS task areas. However, a few of these areas, such as DFHEIBLK or SQLCA, are defined within the COBOL program and, therefore, reside in COBOL storage. For these areas, display past data values by specifying the data name defined in the COBOL program. If the CORE keyword is used (for example, CORE=EIB), current data values are displayed.

Usage Notes

Data monitoring is very CPU intensive and requires a significant amount of storage over and above the storage required by the programs being monitored. Compute the storage requirements as follows:

500 bytes * number of transaction entries as defined by the STMTTRCE option + size of mirror area for each program in the transaction for which data monitoring is active, where the mirror area is equal to the size of storage for the monitored program.

If you should encounter storage problems while using the data monitoring feature, perform one or more of the following actions:

- Reduce the use of data monitoring in the region
- Make the STMTTRCE value smaller in the default options table, IN25OPTS
- Increase the EDSALIM for the CICS region

Display or Modify Auxiliary Storage (FILE)

Use the CA InterTest for CICS FILE facility even when no program is executing to display and modify auxiliary storage. Access this facility from the ITST Primary Option Menu, Option **4 Auxiliary Storage**. The Auxiliary Storage Menu, shown next, appears.

As the Auxiliary Storage Menu shows, this facility enables you to examine or modify files, DB2 and SQL/DS databases, DL/I databases, and CICS temporary storage and transient data records.

```
-----------------   CA InterTest r8.5  AUXILIARY STORAGE MENU ---------------
 OPTION  ===>  1
```

Select an auxiliary storage type, specifying the following optional criteria.

```
     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):

    protsym_ p*_______ ________ ________ ________ ________ ________ ________



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

Note: If you type FILE on the command line of a breakpoint display, you bypass this menu and go to the initial file facility display. To access the Auxiliary Storage menu from a Breakpoint, type **ITST** on the command line; then select option **4** from the Primary Option Menu.

For more information:

Accessing Auxiliary Storage: FILE (see page 441)

FILE at a Breakpoint

Use FILE at a breakpoint to add test records. For example, you can have 12 records in a test file, one for each of 12 test conditions. However, when you get to a breakpoint you realize there is another test condition. To test the condition in the current test run and have a complete test file for future runs, go to FILE, add another test record, return to the breakpoint, and continue program execution so the program reads the new record.

If you are using FILE, you do not have to enter a record from scratch. Retrieve a current record, modify it, and then add it as a new record to the file.

FILE Changes Do Not Affect Main Storage at a Breakpoint

FILE changes do not automatically affect main storage.

Example

If you are stopped at a breakpoint and you use FILE to change the record that your program is processing, the change you made using FILE is not automatically reflected in your program's main storage because your program has already read the record.

To make the change in main storage, return to the breakpoint screen, and do one of the following actions:

- Instruct the program to read the record again
- Repeat the changes to main storage that you made to the record or database

Note: If multiple fields need to changing, select multiple items for display or modification from the Working-storage or Local-Storage section of a COBOL program.

To return to the breakpoint display after using FILE, press **PF3** or **Clear**. (If you also accessed the CA InterTest for CICS CNTL, CORE, or Help facilities, you might need to press Clear more than once.)

Changes to data using FILE are **permanent updates** to the records or database. It is as if you wrote a batch program to perform the updates.

The Breakpoint Primary Option Menu

When you are at a breakpoint, CA InterTest for CICS includes a new Breakpoint Primary Option Menu that places many breakpoint activities right at your fingertips. A sample Breakpoint Primary Option Menu screen follows.

```
    CA InterTest r8.5  BREAKPOINT PRIMARY OPTION MENU
OPTION  ===>

   1  Main menu            Display the CA InterTest primary option menu
   2  Status               Display/remove current monitoring options
   3  Abend                Abend the breakpointed task
   4  Resume               Resume breakpointed task menu options
   5  Override             Override automatic breakpoint default processing
   6  Dump                 Cause dump, resume from next sequential instruction
   7  Disconnect           Disconnect the breakpoint from this terminal
   8  Hogan SMART          Invoke Hogan System's debugging facility
   X  Exit                 Terminate breakpoint menu processing




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

Notice that from the Breakpoint Primary Option Menu, Option **1 Main Menu** takes you to the Primary Option Menu.

Access

To access the Breakpoint Primary Option Menu from any Breakpoint, enter **MENU** on the command line or press **PF6 Menu**.

PF1 Help

To get details on using any of these options, press **PF1** for help.

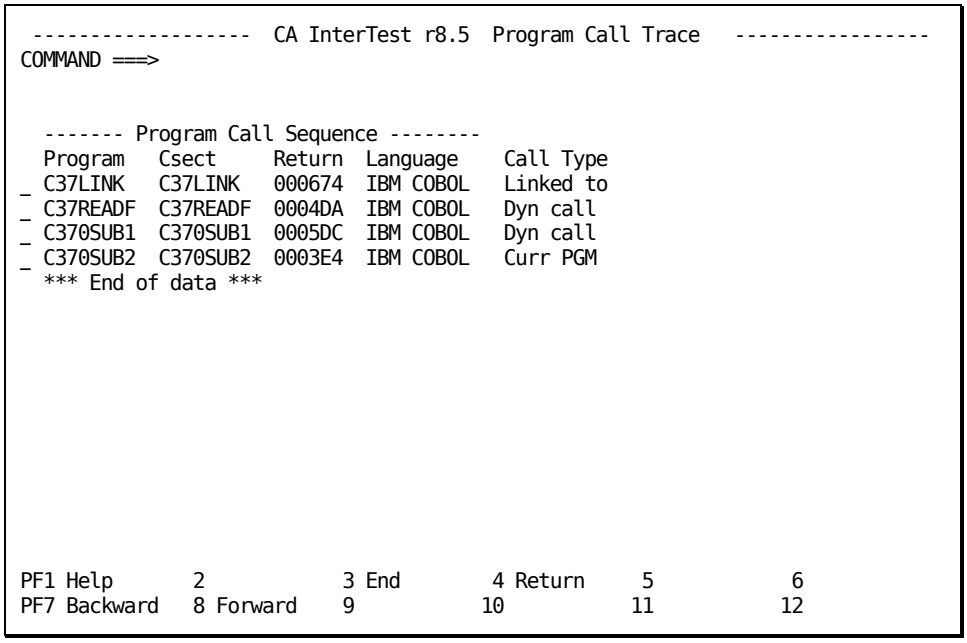
Press **PF3 End** to exit this menu and return to your breakpoint display.

The CALLTRACE Facility

The CALLTRACE facility enables you to view the logic flow of a group of programs. Once the CALLTRACE mode is active view the storage for each program in the same way it is viewed from a breakpoint. Use the CALLTRACE command only if your programs adhere to standard linkage conventions as follows:

- COBOL dynamic or static calls to other COBOL programs
- EXEC CICS LINKED or XCTLed to PROGRAMs
- Standard MVS save area linkage

In the following example the CALLTRACE was issued while program C370SUB2 was stopped at a breakpoint. The CALLTRACE screen shows how program C370SUB2 received control.



```
 ------------------  CA InterTest r8.5  Program Call Trace   -----------------
COMMAND ===>


  ------- Program Call Sequence --------
  Program   Csect     Return  Language    Call Type
_ C37LINK   C37LINK   000674  IBM COBOL   Linked to
_ C37READF  C37READF  0004DA  IBM COBOL   Dyn call
_ C370SUB1  C370SUB1  0005DC  IBM COBOL   Dyn call
_ C370SUB2  C370SUB2  0003E4  IBM COBOL   Curr PGM
  *** End of data ***









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

Select any program in the list having symbolic data by typing an **S** to the left of the program name. The source listing display facility runs in a special mode and positions the listing at the point where it transferred control.

Once the source listing display is in calltrace mode, display program variables the same way as they are for the currently breakpointed program.

For IBM CICS Transaction Server, also display the active channels and containers for any program level by entering a c next to any program and pressing Enter.

The CHANNEL Command

The CHANNEL command enables you to list each of the active channels and view each of the containers within each channel. This feature is only available for IBM CICS Transaction Server.

To display the channels and containers, use one of these two commands:

- Enter the CHANNEL command on the command line from any breakpoint
- Enter a c line command next to any program in the CALLTRACE display

```
---------------  CA InterTest r8.5   CHANNEL/CONTAINER STORAGE  ---------------
COMMAND ===>

  Type S to display main storage.                                      More:

  Channel          Container     Curr CPGID Length    Character Data
_ LNK0CHAN-01      LNK0CONT-00      x 00037 0008106 * 2 LNK0CH02 CONTAI..... *
_                    +000FD8                       * ********************** *
_                    +001FB0                       * ********************** *
_                    +002F88                       * ********************** *
_                    +003F60                       * ********************** *
_                    +004F38                       * ********************** *
_                    +005F10                       * ********************** *
_                    +006EE8                       * ********************** *
_                    +007EC0                       * ********************** *
_ LNK0CHAN-01      LNK0CONT-01      x 00037 0008106 * 1 LNK0CH01 CONTAI..... *
_                    +000FD8                       * ********************** *
_                    +001FB0                       * ********************** *
_                    +002F88                       * ********************** *
_                    +003F60                       * ********************** *
_                    +004F38                       * ********************** *

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

Each entry in the display represents a container, or in the case of larger containers, a segment of a container. The display shows the following information:

Field	Description
Channel	The name of the channel in which the container has been stored
Container	The name of the container, or the segment offset withing the container
Curr	An X indicates that this container was passed to the current program
CPGID	The code page identifier for this container in EBCDIC format
Length	The full length of the container

Field	Description
Character Data	An EBCDIC display of the first twenty-two bytes of each container or segment

To view the storage within a container or segment, place an s on the line next to that container or segment and press **Enter**.

The Storage Facility

While stopped in the source listing facility for the currently breakpointed program enter the STORAGE command on the command line. The following example is a sample Storage facility screen.

```
          ----------------------  CA InterTest r8.5  USER STORAGE  -----------
----------
 COMMAND ===>

   Type S to display main storage.

      Usage:  User24:     3.12KB      CICS24:     0.00KB
              User31:   259.54KB      CICS31:     0.00KB

   Type  Address  (length)  Storage at that location
 _ SU24  00200470 (0000810) 00000043 000004B0 0003D61B 0004B100 .........O.....
 _ SU24  00200000 (0000470) 00B46EC4 C6C8C5C9 E4E24040 40404040 ..>DFHEIUS
 _ SU31  1DC3FE30 (0001000) C8C1D5C3 1DC36FF8 1DC3CCC8 00000000 HANC.C?8.C.H....
 _ SU31  1DC3EE30 (0001000) E2E3D2E4 1DC392DC 1DC39700 00000FF0 STKU.Ck..Cp....0
 _ SU31  1DC3CCC0 (0002170) C8C1D5C3 1DC3FE38 1DC36FF8 00000000 HANC.C...C?8....
 _ SU31  1DC34B20 (00081A0) 00000000 00000000 00000000 00000000 ................
 _ SU31  1DC0BA20 (0029100) C8C1D5C3 1DC05CC8 1DC05CC8 00000000 HANC.{*H.{*H....
 _ SU31  1DC037F0 (0008230) 1DC037C8 1DC061A8 1D3D3BCB 1DD44602 .{.H.{/y.....M..
 _ SU31  1DC00040 (00037B0) 1DC02CE0 00000000 1DC00058 00000000 .{.\.....{......
 _ SU31  1DC00000 (0000040) 4C4CD9E4 E6D76E6E 00000000 00008190 <<RUWP>>......a.
   *** End of data ***


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

Select any piece of storage from the list resulting in a core dump of the selected area.

At the time the screen is built each piece is inspected to see if there is valid leading and trailing crumple zone. If not, a message is produced indicating there is a corrupted storage area, the list shows an * next to the corrupted areas.

The Backtrace Facility

The Backtrace facility lets you view the logic flow of your program. It is designed to answer the question, "How did I get to this point?"

The Backtrace facility enables you to do the following tasks:

- View the Backtrace Summary for a summary of the program's execution path
- View the Source Listing Backtrace for a statement-by-statement look at the program's execution path
- Use special PF keys to trace the program's execution path
- Reposition the source listing to view the program's execution path from different backtrace positions

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 position are available in the Backtrace facility.

Access the Backtrace Facility

To access the Backtrace facility from the source listing facility and to view the Backtrace Summary, enter **BTRACE** on the command line and press Enter, or press **PF11** from the Source Listing Breakpoint screen. CA InterTest for CICS positions the Backtrace Summary at the **last executed statement**, which is the current breakpoint location. To view your program's backtrace statement-by-statement, press **PF2** from the Backtrace Summary to access the Source Listing Backtrace.

Note: While you are using the Backtrace facility, the execution PF keys are disabled and replaced with special forward and backward tracing PF keys.

From the Backtrace Summary, perform the following actions:

- Access the Source Listing Backtrace to view statement-by-statement details of the program's execution path by:
  - Pressing **PF2**
  - Marking a statement block with **S** to indicate a starting position, and 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

The following screen shows the Backtrace Summary.

```
              CA InterTest  BACKTRACE SUMMARY
COMMAND ===>
Program= COBDEMO                                    ❶  From 0001 To 0011 Of 0011

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           10           11 Prev Bloc 12 Next Bloc
-------------------------------------------------------------------------------
 ❷ ❸       ❸         ❹
 S Bkmk       Stmt Block  | Source Listing
 _ ____     #790.0...#791.0|  PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.
 _ ____     #791.0...#801.0|      CALL 'DFHEI1'.
 _ ____     #801.0...#802.0|      CALL 'DFHEI1' USING DFHEIV0
 _ ____     #804.0...#804.0|      IF EIBAID = DFHCLEAR
 s ____     #812.0...#813.0|       MOVE '  Y    i 00557   ' TO DFHEIV0
 _ ____     #813.0...#818.0|      CALL 'DFHEI1' USING DFHEIV0  TSQNAME TASKSTR
 _ ____     #822.0...#823.0|      IF EIBCALEN = 0
 _ ____     #878.0...#881.0|       MOVE '  }           00598   ' TO DFHEIV0
 _ ____     #881.0...#887.0|      CALL 'DFHEI1' USING DFHEIV0  DFHC0070 DFHEICB
 _ ____     #887.0...#888.1|      CALL 'DFHEI1' USING DFHEIV0.
 * ____     #897.0... Bkpt |      ADD +1 TO TASKNUM.

❺ CAIN2988 First and Last Backtrace Stmt Block
```

❶

From field indicates the relative backtrace statement blocks displayed on the screen. There are a maximum of 580 backtrace statement blocks that record a program's execution path. The oldest statement blocks are re-used as needed.

To reposition the Source Backtrace Summary, overtype the information in the From field with a relative statement block number or with a one- to four-byte user-specified tag or bookmark.

❷

Select (S) column positions a display of the Source Listing Backtrace at a selected backtrace statement. Enter **S** next to any entry and press Enter. The Source Listing Backtrace displays from the specified backtrace position. Only **one S** is allowed.

Note: An * in the Select column indicates the current backtrace position.

❸

Bookmark (Bkmk) column is used to assign a unique tag or ID to one or more backtrace positions.

- Specify multiple bookmarks.
- Each bookmark must contain at least one non-numeric character. For example, P103, UB7, T1, A, and so on.
- Each bookmark must be unique.
- Once a bookmark is assigned to a **specific backtrace position**, it remains attached to that backtrace position until the bookmark is replaced or removed by subsequent program execution.
  - There are 580 backtrace positions, and the oldest entries are replaced.
  - If a backtrace position is not replaced between breakpoints, use the bookmark to reposition your Backtrace Summary by specifying the **bookmark name** in the From field of the Backtrace Summary screen.

❹

Statement (Stmt) Block column identifies a block of contiguously executed statement numbers.

- Each line entry is a pair of program locations identifying the first and last statement numbers or offsets for a piece of sequentially executed code.
- For COBOL programs, the decimal value of the statement number indicates a specific verb in a statement. The first verb is 0, the second is 1, and so on.
- The top left entry was executed first, and the bottom right entry was executed most recently.
- After each statement number or offset in the right column there was a break in sequential execution, such as a CALL to CICS or a branch to another location in the program. Each break results in a new line entry.
- Plus signs indicate offsets; for example, +204.

❺

Source Listing Summary column displays the source code associated with the first statement executed in the statement block, which is identified by the statement number in the left column. The left margin default is 01 for COBOL and PL/I programs, and 34 for Assembler programs.

Note: For display purposes, the starting column is either the left margin default or the Margin= value that was specified on the Source Listing Breakpoint screen, whichever is greater.

❻

Message Line displays any messages about the section of backtrace that you are currently viewing.

The Backtrace Summary

The Backtrace Summary summarizes the program's execution path using statement blocks. The statement block column is located on the left side of the screen and contains two columns of numbers:

- Statement numbers in the left column **receive control** from a branch, a command level CALL, a PERFORM, or a GOTO
- Statement numbers in the right column **issue** the branch to the next statement identified

By tracing the execution path, block-by-block, follow the flow of your program's logic.

Example

To use the information in the statement block columns, first look at the last entry, as in the previous screen:

```
#897.0... Bkpt
```

This block answers the question: How did I get to the current breakpoint? In this case, the program entered statement 897 and executed all of the machine instructions up to the current breakpoint.

The next question you might ask is: How did statement 897 receive control? To answer this question, you must look at the preceding statement block:

```
#887.0...#888.1
```

The statement number in the right column, 881.1, is the statement that caused a branch to statement 897.

Note: You might see a statement branch to itself, as in the following example from the previous figure:

```
   #812.0...#813.0 ←
➔  #813.0...#818.0
```

This happens when a CICS command is issued and execution was returned to the same statement.

Backtrace Summary Screen PF Key Assignments

The first Backtrace Summary screen always displays the most recently executed backtrace entries at the bottom of the backtrace table. Use the From field or the following PF keys to navigate though the Source Backtrace Summary.

PF Key	Function
PF1 Help	Displays online Help for the current screen.
PF2 Backtrace	Redisplays the Source Listing Backtrace screen from the current backtrace position. The current backtrace position is indicated by an asterisk (*).
PF3 End	Displays the Source Listing Breakpoint screen at the last breakpoint.
PF5 1st Stmt	Repositions the Backtrace Summary to the first (oldest) entry in the Backtrace.
PF6 Last Stmt	Repositions the Backtrace Summary to the last (newest) entry in the Backtrace.
PF7 Backward	Scrolls the Backtrace Summary backward (up) a full screen.
PF8 Forward	Scrolls the Backtrace Summary forward (down) a full screen.
PF11 Prev Bloc	Displays the previous statement block.
PF12 Next Bloc	Displays the next statement block.

Note: A column of backtrace statement blocks is available on the Detailed Breakpoint screen. For more information, the Detailed Breakpoint screen (see page 178).

The Source Listing Backtrace

To switch to the Source Listing Backtrace and to view the backtrace statement by statement, press **PF2** or type S next to a statement block in the Select column of the Backtrace Summary to select a starting position and press **Enter**, as shown in the previous screen.

Note: In the following screens, gray shading indicates highlighted statements.

The following screen shows the Source Listing Backtrace.

```
CA InterTest  PROTDEM FILE SOURCE LISTING BACKTRACE
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 Summary   3 End       4 Profile    5 1st Stmt   6 Last Stmt
     7 Backward  8 Forward   9 Prev Stmt 10 Next Stmt 11 Prev Bloc 12 Next Bloc
========> Backtrace at #00812  (00812 > 00813 executed     1 times) <=========
   00810*          ITEM(TSQITEM)
   00811*          ENDEXEC.
 _ ====>      MOVE '  Y    i 00557   ' TO DFHEIV0
 _ 00813     CALL 'DFHEI1' USING DFHEIV0  TSQNAME TASKSTRUCTURE TSQLEN
   00814     DFHDUMMY TSQITEM.
   00815
   00816
   00817
 _ 00818     IF EIBAID = DFHPF2
 _ 00819      OR EIBAID = DFHPF14
 _ 00820      OR TASKSWITCH = DFHPF2
 _ 00821         GO TO EXPANDEDDEMO.
 _ 00822     IF EIBCALEN = 0
 _ 00823         GO TO SENDFIRSTSCREEN.
 _ 00824     GO TO CONTINUETASK.
```

Read the Source Listing Backtrace

To clearly specify the program's execution path statement-by-statement, source statements executed **from the current backtrace position** are highlighted. However, when you specify a non-backtrace Source Listing option or use PF7 or PF8 to reposition the Source Listing Backtrace, the highlighting of executed statements is temporarily suspended until a backtrace PF key 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 on the previous line indicates:

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

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

- ====> indicates the first statement in a statement block
- <==== indicates the last statement in a statement block
- ===== indicates all other statements within a statement block

Source Listing Backtrace PF Keys

Trace your program's execution path on the Source Listing Backtrace display using the following PF keys:

PF Keys	Function
PF1 Help	Initiates the CA InterTest for CICS online help facility.
PF2 Summary	Displays the Backtrace Summary.
PF3 End	Terminates the backtrace session and redisplays the Source Listing Breakpoint screen at the last breakpoint.
PF4 Profile	Accesses the Source Listing Profile, where you set source listing features and PF9 and PF10 stepping amount settings.
PF5 1st Stmt	Repositions screen to the first (oldest) entry in the backtrace.
PF6 Last Stmt	Repositions screen to the last (newest) entry in the backtrace.
* PF7 Backward	Scrolls screen backward (up) a full screen.
* PF8 Forward	Scrolls screen forward (down) a full screen.
PF9 Prev Stmt	Repositions the Source Listing Backtrace to the previously executed statement.
PF9 Prev *nnn*	Automatically traces backward the number of statements set in the Stepping Amount= field on the Source Listing Profile (PF4). Stepping Amount value is *nnn*.
PF10 Next Stmt	Repositions the Source Backtrace to the next statement.
PF10 Next *nnn*	Automatically traces forward the number of statements set in the Stepping Amount= field on the Source Listing Profile (PF4). Stepping Amount value is *nnn*.
PF11 Prev Bloc	Displays the previous statement block.
PF12 Next Bloc	Displays the next statement block.

*PF keys 7 and 8 temporarily suspend backtrace highlighting until PF5, 6, 9, 10, 11, or 12 are specified.

Navigate Through Program Execution

Trace your program's execution path by performing the following actions:

- Using PF5, PF9, and PF11 to go backward
- Using PF6, PF10, and PF12 to go forward
- Setting the Source Listing Profile **Stepping Amount** and **Wait Amount**, and then using PF9 and PF10 to automatically trace the execution backward or forward

When you first access the Source Listing Backtrace by pressing PF2 on the Backtrace Summary screen, the following screen appears. Access this screen also from any other section of backtrace by pressing **PF6**, which positions the backtrace at the last statement executed (the current breakpoint).

```
             CA InterTest  PROTDEM FILE SOURCE LISTING BACKTRACE
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 Summary   3 End        4 Profile    5 1st Stmt   6 Last Stmt
     7 Backward  8 Forward   9 Prev Stmt 10 Next Stmt 11 Prev Bloc 12 Next Bloc
========> Backtrace at #00897  (00897 > curr. executed     1 times) <=========
 _ 00895 CONTINUETASK.
   00896**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
 A   ==>     ADD +1 TO TASKNUM.
 _ 00898     IF TASKNUM = 1
 _ 00899         MOVE 'DMAPASR'  TO MAPNAME.
 _ 00900     IF TASKNUM = 2
 _ 00901         MOVE 'DMAPSUM'  TO MAPNAME.
 _ 00902     IF TASKNUM GREATER 2
 _ 00903         GO TO SENDENDMSG.
 _ 00904     GO TO REWRITETSQ.
 _ 00905 REWRITETSQ.
   00906*EXEC CICS WRITEQ TS
   00907*          REWRITE
   00908*          QUEUE(TSQNAME)
   00909*          FROM(TASKSTRUCTURE)
```

To reposition the source listing to the first statement of the current statement block or to the beginning of the previously executed statement block, press PF11 Prev Bloc.

Example

If **PF11** was pressed from the Source Listing Backtrace screen shown previously, the following screen appears.

```
CA InterTest  PROTDEM FILE SOURCE LISTING BACKTRACE
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 Summary   3 End        4 Profile    5 1st Stmt   6 Last Stmt
     7 Backward  8 Forward   9 Prev Stmt 10 Next Stmt 11 Prev Bloc 12 Next Bloc
========> Backtrace at #00897  (00897 > curr. executed     1 times) <=========
     00885*          ENDEXEC.
   _ 00886      MOVE ' 00604   ' TO DFHEIV0
   _ ====>      CALL 'DFHEI1' USING DFHEIV0.
   _ 00888     IF EIBAID = DFHENTER GO TO CONTINUETASK.
   _ 00889     IF EIBAID = DFHCLEAR GO TO SENDENDMSG.
   _ 00890     IF EIBAID = DFHPF3   GO TO SENDENDMSG.
   _ 00891     IF EIBAID = DFHPF15  GO TO SENDENDMSG.
   _ 00892     IF EIBAID = DFHPF2   GO TO EXPANDEDDEMO.
   _ 00893     IF EIBAID = DFHPF14  GO TO EXPANDEDDEMO.
   _ 00894     GO TO SENDFIRSTSCREEN.
   _ 00895 CONTINUETASK.
     00896**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
  A    ==>     ADD +1 TO TASKNUM.
   _ 00898     IF TASKNUM = 1
   _ 00899         MOVE 'DMAPASR'  TO MAPNAME.
```

To reposition the source listing to the beginning of the next backtrace statement block, press **PF12** Next Bloc.

Example

If you pressed **PF12** from the Source Listing Backtrace shown previously, the following screen appears.

```
CA InterTest  PROTDEM FILE SOURCE LISTING BACKTRACE
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 Summary   3 End        4 Profile    5 1st Stmt   6 Last Stmt
     7 Backward  8 Forward   9 Prev Stmt 10 Next Stmt 11 Prev Bloc 12 Next Bloc
========> Backtrace at #00897  (00897 > curr. executed     1 times) <=========
     00885*          ENDEXEC.
   _ 00886      MOVE ' 00604   ' TO DFHEIV0
   _ ====>      CALL 'DFHEI1' USING DFHEIV0.
   _ 00888     IF EIBAID = DFHENTER GO TO CONTINUETASK.
   _ 00889     IF EIBAID = DFHCLEAR GO TO SENDENDMSG.
   _ 00890     IF EIBAID = DFHPF3   GO TO SENDENDMSG.
   _ 00891     IF EIBAID = DFHPF15  GO TO SENDENDMSG.
   _ 00892     IF EIBAID = DFHPF2   GO TO EXPANDEDDEMO.
   _ 00893     IF EIBAID = DFHPF14  GO TO EXPANDEDDEMO.
   _ 00894     GO TO SENDFIRSTSCREEN.
   _ 00895 CONTINUETASK.
     00896**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED
  A    ==>     ADD +1 TO TASKNUM.
   _ 00898     IF TASKNUM = 1
   _ 00899         MOVE 'DMAPASR'  TO MAPNAME.
```

Ending a Source Listing Backtrace Session

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

To turn off the backtrace display mode, press Clear or **PF3**. CA InterTest for CICS redisplays the Source Listing Breakpoint screen positioned at the last breakpoint.

Additional Breakpoint Displays

The Breakpoint Information screen and the Detailed Breakpoint screen provide you with more information than the Source Listing Breakpoint screen. However, you cannot test from the Breakpoint Information screen, and the Detailed Breakpoint screen requires you to specify most functions by completing another menu or entering single-line commands.

Look at the Breakpoint Information screen from any Source Listing Breakpoint screen:

- When you are at a Source Listing Breakpoint, press PF3 DET BKPT to look at the Breakpoint Information screen.
- To return to the Source Listing Breakpoint screen, press PF4 Source.

Note: If the Source List BKPT= ON, CA InterTest for CICS displays the Source Listing Breakpoint version. If the Source List BKPT= OFF, CA InterTest for CICS displays the Detailed Breakpoint version.

The following screen shows the Breakpoint Information screen when Source List BKPT=ON. The format does not include lines of the program listing.

```
Program COBDEMO stopped by InterTest in statement 882.0 (Offset +01A8E)     ❶
 Automatic breakpoint; ASRA abend (0C7) detected and prevented.

     Caused by invalid decimal arithmetic data format.
                                                                  ❸ Backtrace
❷Tran= DEMC from term L6D1 monitored as PROG=COBDEMO               #776.0   #786.0
DFHCOMMAREA does not exist.                                       #786.0   #787.0
                                                                  #789.0   #789.0
                                                                  #797.0   #798.0
                                                                  #786.0   #787.0
                                                                  #814.0   #815.0
                                                                  #815.0   #816.0
Depress PF3 for Source Listing Display.                           #821.0   #823.0
                                                                  #823.0   #833.0
                                                                  #833.0   #838.0
                                                                  #838.0   #839.0
                                                                  #841.0   #851.0
❹The two fields shown are at address 01C2A4 and 2B817C:          #851.0   #866.0
000000                                *...            *           #866.0   #872.0
1C                                    *.              *           #872.0   #873.1
                                                                  #882.0 to here.
❺==> CNTL=GO,TASK=00290,A                                               Margin:

 ❻Overtype the above or use a PF key or press ENTER to abend without a dump.

 1 Help       2            3 Source      4           5 Resume      6 Menu
 7            8            9 1 instruc  10 1 verb   11 Backtrace  12 Status
```

Note: The information displayed on the Breakpoint Information screen is also available on the Detailed Breakpoint screen, which is discussed in the Detailed Breakpoint screen later in this chapter.

All fields except the command line are protected.

❶

Identifies the statement or offset location where the breakpoint occurred in your program. For COBOL programs, the decimal value of the statement number indicates which verb halted execution. The first verb is 0, the second verb is 1, and so on.

❷

Identifies the transaction and terminal ID where the program is executing, and how monitoring was requested, such as for a program, transaction, or terminal.

❸

Shows the backtrace as described in the section Tracing Program Execution. Additional backtrace entries are accessed using PF11. To return, press **Clear**.

❹

Indicates, in hexadecimal and character format, the address and contents of main storage relevant to the breakpoint.

❺

Displays the command line with a CNTL default command. Override this command with any other CA InterTest for CICS command or transaction ID, such as HELP or CORE. It is followed by a line that indicates any errors or messages. Enter also a fast-path command to jump to a menu, such as =1.2.1 to go to the Program Monitoring menu.

❻

Lists the PF keys that are available from the Breakpoint Information screen. For details, detailed breakpoint PF keys.

The Detailed Breakpoint Screen

The Detailed Breakpoint display is designed for experienced CA InterTest for CICS users with knowledge of CA InterTest for CICS commands and CICS. It contains more diagnostic information than the Source Listing Breakpoint display. It is displayed when the Source Listing Profile setting Source List BKPT=OFF.

From the Detailed Breakpoint display access the Breakpoint Primary Option Menu, but you cannot use the one-character commands that are available on the Source Listing Breakpoint display, such as **m** to modify or **d** to display. Instead, access the Main Storage Menus (fast-path entry is =1.3) or enter a one-line command in the command field on the bottom of the screen.

A Detailed Breakpoint screen appears when Source List BKPT=OFF. A sample screen follows.

```
Program COBDEMO stopped by InterTest in statement 882.0 (Offset +01A8E)❶
❷ Automatic breakpoint; ASRA abend (0C7) detected and prevented.

       Caused by invalid decimal arithmetic data format.

                                                        ❹    ❻ Backtrace
❸Tran= DEMC from term L6D1 monitored as PROG=COBDEMO     🡻
#776.0  #786.0
❺DFHCOMMAREA does not exist.                                    #786.0  #787.0
                                                                #789.0  #789.0
  00879     GO TO SEND FIRST SCREEN.                            #797.0  #798.0
  00880 CONTINUE TASK.                                          #786.0  #787.0
  00881**** TASKNUM *NOTE* FIELD MUST BE INITIALIZED             #814.0  #815.0
  00882     ADD +1 TO TASKNUM.                                  #815.0  #816.0
❼00883     IF TASKNUM = 1                                       #821.0  #823.0
  00884         MOVE 'DMAPASR'  TO MAPNAME.                     #823.0  #833.0
  00885     IF TASKNUM = 2                                      #833.0  #838.0
  00886         MOVE 'DMAPSUM'  TO MAPNAME.                     #838.0  #839.0
                                                                #841.0  #851.0
❽ The two fields shown are at address 033DA4 and 2AE17C:         #851.0  #866.0
  000000                                *...             *      #866.0  #872.0
  1C                                    *.               *      #872.0  #873.1
                                                                #882.0 to here.


❾==> CNTL=GO,TASK=00378,A                                              Margin:

❿Overtype the above or use a PF key or press ENTER to abend without a dump.

 1 Help       2           3 Source      4           5 Resume      6 Menu
 7            8           9 1 instruc  10 1 verb   11 Backtrace  12 Status
```

Note: All fields except the command line are protected.

❶

Identifies the statement or offset location where the breakpoint occurred in your program. For COBOL programs, the decimal value of the statement number indicates which verb halted execution. The first verb is 0, the second verb is 1, and so on.

❷

Indicates what kind of breakpoint occurred. For a review of what each letter means, see the Examine the Source Listing Breakpoint screen (see page 136) at the start of this chapter. If an automatic breakpoint occurred, the reason for it follows.

❸

Identifies the name of the transaction and terminal ID where the program is executing and how monitoring was requested (such as for a program, transaction, or a terminal.).

❹

For PL/I and Assembler programs, this area displays the machine instruction that triggered the breakpoint, the machine instruction that last executed, and its offset or absolute address. (This is not shown in this screen.)

❺

Indicates the COMMAREA status and size.

❻

Shows the backtrace, which is described in the section Tracing Program Execution. Additional backtrace entries are accessed using PF11.

❼

Displays source listing code, **only** if you:

- Saved the listing in the CA InterTest for CICS symbolic file
- Chose Source List BKPT=OFF on your Source Listing Profile screen. Without this option specified, CA InterTest for CICS does not display this screen at all. The Breakpoint Information screen displays instead.

❽

Indicates, in hexadecimal and character format, the address and contents of main storage relevant to the breakpoint.

❾

Displays the command line with a default CNTL command. Override this command with a CA InterTest for CICS command or transaction ID. It is followed by a line that indicates any errors or messages.

❿

Lists the PF keys that are available from the Detailed Breakpoint Display screen.

For more information:

CNTL Commands and Menus (see page 295)
Accessing Main Storage: CORE (see page 375)

Read the Backtrace on the Detailed Breakpoint Screen

Use the Source Listing Facility Backtrace feature to examine the path your program took during execution, and to help solve logic problems. On the Detailed Breakpoint screen, only one column at-a-time of backtrace entries is available. To access additional backtrace entries, press PF11 or switch to the Source Listing Breakpoint display mode for a more detailed backtrace listing.

The backtrace shows the path of the most recently executed code.

- The top left entry was executed first, and the bottom right entry was executed last.
- Each line entry is a pair of program locations identifying the first and last statement numbers or offsets for a piece of sequentially executed code.
- The decimal value after statement numbers for COBOL code indicates which verb in the statement was executed last. The first verb is given a decimal value of 0, the second 1, and so on.
- After each statement number or offset in the right column there is a break in sequential execution, such as a CALL to CICS or a branch to another location in the program. Each break results in a new line entry.
- If a program loops through a section of code more than once, the backtrace displays the number of times the loop was executed below the line entry, instead of repeating the execution sequence.
- Plus signs indicate offsets; for example, +204.

To return to the breakpoint display, press **PF3** or **Clear**.

The detailed breakpoint PF keys follow.

PF Key	Function
PF1 Help	Initiates the CA InterTest for CICS online help facility.
PF2	Unassigned.
PF3 Source	Displays the Source Listing version of the breakpoint.
PF4	Unassigned.
PF5 Resume	Resumes execution at the next instruction using the amount set on PF10.
PF6 Menu	Displays the Breakpoint Primary Option Menu.
PF7	Unassigned.
PF8	Unassigned.
PF9 1 Instruc	Executes the next machine instruction and then stops.

PF Key	Function
PF10 001 Verb PF10 001 Stmt PF 10 Registers	For COBOL, executes the next COBOL verb and then stops. For PL/I, executes the next PL/I statement and then stops. For Assembler, toggles registers on and off the screen.
PF11 Backtrace	Displays the column of backtrace available at this breakpoint.
PF12 Status	Displays a monitoring status report for the monitored entry (program, transaction, or terminal) that caused the breakpoint.
PF22 Shift Left	Shifts source code left; COBOL and PL/I programs shift 10 characters at a time; Assembler programs shift to 02, 40, 50, and then 61. If you have only 12 PF keys, use the Margin= field to shift your source code.
PF23 Shift Right	Shifts source code to the right the same amounts as PF22.

Continue Execution

To continue execution from the Detailed Breakpoint or the Breakpoint Information screen, perform these steps:

- Press PF9 or PF10 to single-step.
- Press PF5 Resume to continue execution starting with the next instruction.
- Press PF6 Menu to access the Breakpoint Primary Option Menu, where you abend, override an automatic breakpoint, step, or access the Resume Menu for additional execution options. For details, The Breakpoint Primary Option Menu (see page 163).
- Enter a CNTL command on the command line.
- Enter GO, NEXT, or RUN on the command line.

Single-Stepping

Press **PF9** or **PF10** to execute a single machine instruction, COBOL verb, or PL/I statement, as indicated in the PF key definition on the bottom of the screen. Use this method to see what data values are being moved into main storage.

Notes:

- The step amount for PF9 and PF10 cannot be changed. However, select alternative step amounts by entering CNTL commands or by using the Resume Menu options. Access the Resume menu using PF6, and select Option **4 Resume** (the fast-path command is **=4**).

- When you use PF10 to single-step through nested COBOL IF statements, it might appear that CA InterTest for CICS is not executing one verb at a time. This is a function of how the compiler generates the object code. To follow closely the execution of nested IFs, use PF9 (execute 1 instruction) instead.

Resume Execution

Access the Resume Menu by pressing **PF6** from the Detailed Breakpoint screen, then selecting Option 4 Resume. Alternatively, jump to the Resume menu by entering =4 in the command line on the Detailed Breakpoint screen.

Using the Resume Menu, perform the following functions:

- Resume program execution until the next breakpoint occurs or the transaction ends
- Resume program execution from a new location, such as a COBOL paragraph name, Assembler or PL/I label, COBOL or PL/I statement number or offset
- Move or disable the current breakpoint
- Execute a specified number of PL/I instructions or COBOL verbs before stopping
- Abend the transaction with or without a dump

For more information:

Resume and Execution Options (see page 213)

CNTL Commands

CA InterTest for CICS provides a default command at each breakpoint display, as shown following:

- The default command at an automatic breakpoint is:

  CNTL=GO,task=xxxxx,A

  A means abend the task without a dump.

- The default command at other breakpoints is:

  CNTL=GO,task=xxxxx,C

  C means continue execution from where you left off.

  To execute the default command, press **Enter**.

To modify the default command, overtype the existing command and press **Enter**.

For more information:

CNTL Commands and Menus (see page 295)

Continue Processing During Breakpoint Processing

To continue processing from either the current line or a specified line number or label until the next breakpoint is encountered, an abend occurs, or the program runs to normal completion, enter:

```
GO line-number | label
```

Line-number

Indicates any line number within an ASSEMBLER, COBOL, or PL/1 program from which you want the execution to continue.

Default: The current statement.

Label

Indicates any valid paragraph or procedure label within an ASSEMBLER, COBOL, or PL/1 program (up to 31 characters in length) from which you want the execution to continue.

Default: The current statement.

Note: GO is available only during breakpoint processing.

Continue Processing During Breakpoint Processing Until Specified Verbs Are Executed

Important! The NEXT command replaces the obsolete STEP command.

```
NEXT verb-count
```

Verb-count

Indicates number of verbs to execute before the next display of the Breakpoint Display screen.

Default: Value of Stepping Amount in Profile.

Note: NEXT is available only during breakpoint processing.

INTC--Continue Processing Until a COBOL PERFORM Has Been Completed Avoiding Breakpoints

NEXT OVER

Specifies that the NEXT command is used to execute through a verb,instruction, or statement and return to that line. This is to avoid breakpoints.

INTC--Continue Processing After a COBOL Program Breakpoint Until the Next Call Or PERFORM Statement

NEXT RETURN

Specifies that the NEXT command is used to continue execution until the next CALL or PERFORM verb. Execution stops when it hits a CALL or PERFORM execution.

Continue Processing During Breakpoint Processing Ignoring Preset Breakpoints

To continue processing from the current statement ignoring all preset breakpoints until an abend occurs or the task runs to normal completion, enter:

RUN

Note: RUN is available only during breakpoint processing.

Continue Processing During Breakpoint One Verb at-a-Time

To continue processing from the current statement one verb at a time displaying the Breakpoint Display screen for a specified time interval until either a specified number of verbs have been executed or the next "CALL" statement is encountered, enter:

AUTOSTEP *interval stop-count*

Interval

Indicates number of seconds (between 1 –59) to display each Breakpoint Display screen.

Default: Wait Amount in Profile.

Stop-count

Indicates number of verbs to execute before stopping.

Default: Stop value in Profile.

Note: AUTOSTEP is available only during breakpoint processing.

See More Code on the Detailed Breakpoint Display

While viewing the Detailed Breakpoint display, you might not see all of the program's source code on your screen. To see more code, perform the following steps:

1. Tab to the Margin= field, enter a number from 0 to 99 to indicate the starting position of the left margin, and press Enter.

   CA InterTest for CICS redisplays the source code beginning in the position you requested. A Margin= value of 0 begins the display from position 1.

2. Press **PF22** to shift the source code left. Press **PF23** to shift the source code right. COBOL and PL/I programs shift 10 characters at a time; Assembler programs shift to positions 02, 40, 50, and then 61.

Restore the Detailed Breakpoint Display

Once you leave the Detailed Breakpoint screen (for example, by pressing PF3 to view the source listing), restore the Detailed Breakpoint display by pressing **PF3** or **Clear**. If you accessed a CA InterTest for CICS facility such as CNTL, CORE, HELP, or FILE, you might need to press PF3 or Clear more than once to restore the breakpoint display.

Switch Between Breakpoint Screens

If the first CA InterTest for CICS breakpoint screen you see is a Detailed Breakpoint display, you might want to switch to the Source Listing Breakpoint Display version for your program testing. To switch from the Detailed Breakpoint screen to the Source Listing Breakpoint screen, perform these steps:

1. Press **PF3** Source. You should now see the Source Listing Breakpoint Display.
2. Enter **PROFILE** in the command line and press Enter. The Source Listing Profile screen appears.
3. Enter **ON** in the Source List BKPT= field and press Enter. You are returned to the Source Listing Breakpoint display.

Once you complete these steps, CA InterTest for CICS displays the Source Listing Breakpoint version at future breakpoints in your test session. For more information, see the chapter "Source Listing Facility."

To switch from the Source Listing Breakpoint to the Detailed Breakpoint display, perform these steps:

1. Enter **PROFILE** in the command line of a Source display and press Enter. The Source Listing Profile appears.
2. Enter **OFF** in the Source List BKPT= field and press Enter.
3. Press **PF3** DET BKPT. You should now see the Detailed Breakpoint display version.

Note: The default value of the Source List BKPT= field is a system-wide, CA InterTest for CICS installation option. To change the default value, have your System Administrator consult the *Installation Guide.*

Indirect Commands

The CA InterTest for CICS indirect commands facility enables you to define a set of commands that execute automatically, or *indirectly,* at a predetermined location in a monitored COBOL or PL/I program. Invoke these commands from an unconditional, conditional, or request breakpoint.

In effect, indirect commands are executed at breakpoints within your program just as other CA InterTest for CICS line commands issued from the Source Listing Display, such as MOVE and GO, are used to alter the flow of your program's logic. The difference is that indirect commands are executed automatically without issuing a breakpoint display. Once the indirect commands are performed, your program continues to execute normally.

Indirect commands save you from having to recompile a program or manually correct an area of code every time it is encountered during a debugging session. With the indirect commands facility, you can perform 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

  Note: Indirect commands are **not** available for Assembler programs.

How Can You Use Indirect Commands?

Use indirect commands to correct the following sample program problems:

- A variable is not being initialized during processing. Using indirect commands initialize the variable every time the program is run and continue execution without recompiling.
- A new routine is needed in your program. Insert the code using indirect commands and continue executing your program without recompiling.

- A section of code is bad. To continue executing or testing without recompiling, write and invoke indirect commands to direct CA InterTest for CICS to go around the bad source code or execute a new series of indirect command statements.
- You must debug a COBOL or PLI program, which is invoked by a web process or run as background task and is not associated with a specific terminal or userID. You must specify your own criteria when the programs breakpoints should be taken or ignored.

Overview of Steps

To invoke indirect commands in your program, perform these steps:

1. Enter the statement numbers and indirect command statements on the Indirect Commands screen. To access the Indirect Commands screen, use Option #11 from a Source Listing screen, or select the Command option on the Program Monitoring menu.
2. Use the Monitoring Menus or Source Listing facility to set a breakpoint (unconditional, conditional, or request) where the indirect commands will take effect. **As you set the breakpoint, you must access the Breakpoint Options menu and specify the first statement number of the Indirect Command as an option to that breakpoint.** Option #12 (Bkpt Opt) on the Source Listing screen provides access to the Breakpoint Options menu.
3. Specify the terminal ID (terminal ID, .ANY, or .NO) or user ID for the indirect commands to take effect.

   The following section describes each of these steps in detail.

Note: Set the breakpoint and give the starting indirect command number before you code the indirect commands. The order is up to you. However, new users often find it easier to code the indirect commands first, and then attach them to breakpoints.

Set a Breakpoint to Execute an Indirect Command

Before accessing the indirect commands facility, first set an unconditional, conditional, or request breakpoint that includes an option to execute an Indirect Command statement number.

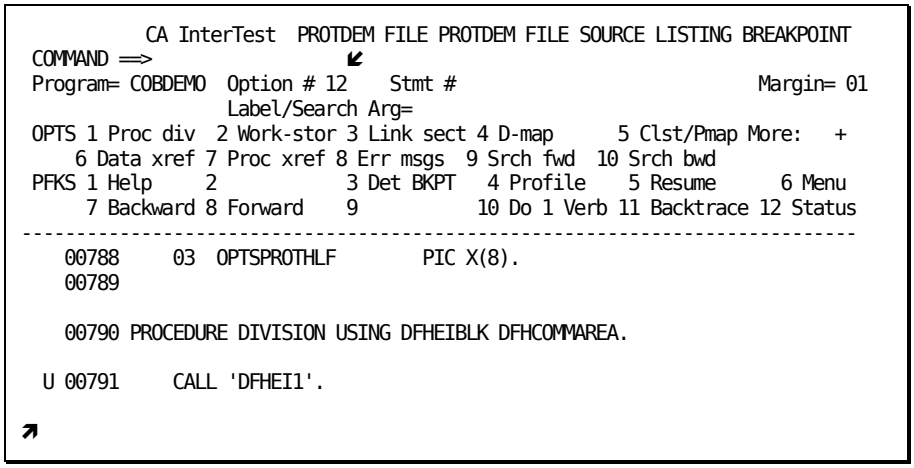
Follow the steps listed next when setting an unconditional or conditional breakpoint from the Source Listing Breakpoint:

1. From the Source Listing Breakpoint, set an unconditional or conditional breakpoint that will invoke the indirect commands.

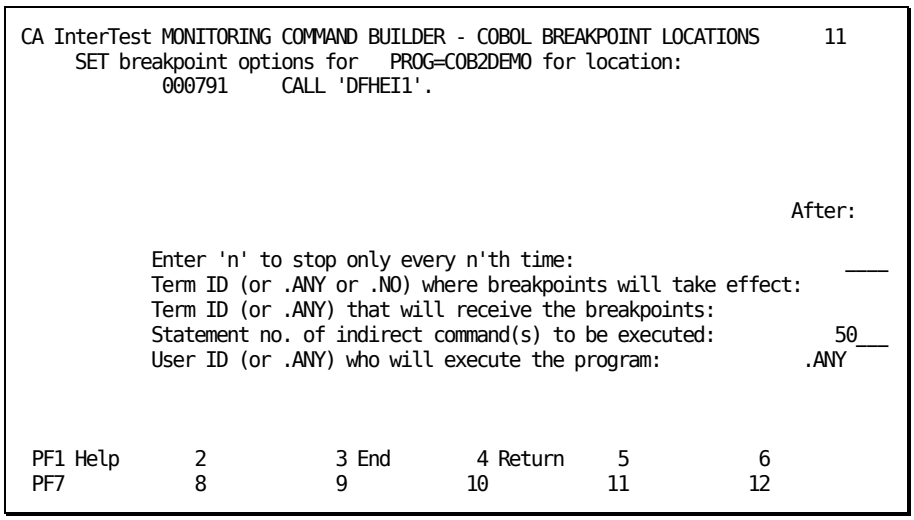
   The following table summarizes how to set the breakpoints from the Source Listing Facility:

Breakpoint	Source Listing Entries
Unconditional	Specify: Option # **12** (Bkpt Opts) **U** to the left of one or more statements Press Enter
Conditional	Specify: Option # **12** (Bkpt Opts) **C** to the left of one or more statements Press Enter
Request	Use Monitoring Menus **Note:** To jump to the Program Monitoring Menu from a breakpoint display, enter **=1.2.1** on the command line.

The following screen shows how to set an unconditional breakpoint for indirect commands from the Source Listing facility. Notice the required Option # 12 entry at the top of the screen.



2. Enter Option # **12** and **U** on Source Listing for Indirect Commands.
3. After pressing Enter, CA InterTest for CICS displays the Breakpoint Locations menu with the location you selected on the Source Listing already indicated.
4. On the Breakpoint Locations menu, specify the (still to be written) one- to five-digit statement number of the first indirect command that you want to be executed at the breakpoint you are setting. For example, enter 10 or 50.



5. Press Enter. CA InterTest for CICS processes the breakpoint as usual, and displays a message to that effect.

   CA InterTest for CICS displays the following Indirect Commands screen.

```
  CA InterTest MONITOR COMMAND BUILDER - COBDEMO  INDIRECT COMMANDS    24

❶   Term ID (or .ANY or .NO): xxxx               ❷Delete ALL: NO
   ❸     ❹

  LINE  COMMAND









❺  1 Help       2           3 End       4          5              6
    7 Backward   8 Forward    9 Top       10 Bottom   11             12
❻ CAIN1255 No indirect commands exist for this program/terminal.
```

❶

Term ID indicates the terminal where the specified indirect commands execute. This value must match the terminal ID specified at the breakpoint that invokes these indirect commands.

xxxx—indicates a specific terminal.

.ANY—indicates that these commands will execute at all terminals.

Note: If indirect commands are specified for both .ANY and your specific terminal, the indirect commands specified for **your** terminal take precedence.

.NO—indicates that these commands execute with no terminal.

❷

Delete ALL provides a quick method to delete all previously specified indirect commands for the program you are testing. Enter one of the following options:

NO—indicates that the indirect commands at the specified terminal will not be deleted (default).

YES—indicates that the indirect commands at the specified terminal will be deleted.

❸

Line Indicates the one- to five-digit sequence number, from 1 to 99999, of the indirect command, which **must** be followed by at least one blank space. This number must be the first item entered except for continued commands, and must begin in the first position of the input line.

❹

Command displays the actual command text to be performed during Indirect Command execution from a breakpoint. For a complete listing of indirect commands, see Indirect Command Syntax.

❺

PF Keys let you scroll through your indirect commands, invoke the Help facility, or exit the function.

❻

Message Line displays messages about your indirect commands.

Code Indirect Commands

Use the Indirect Commands screen to code all indirect commands for your program. See the end of this chapter for syntax information.

When you are finished coding an indirect command statement, perform these steps:

1. Press Enter. CA InterTest for CICS translates everything you entered into uppercase characters and checks the command for syntactical accuracy. If the command is correct, one of the following messages is displayed:

   ```
   CAIN1261 Indirect command(s) added.
   Indirect command(s) replaced.
   ```

2. When you have coded all of your indirect commands, press **PF3** or **Clear.**

From Source Listing

If you are working from the Source Listing facility, you return to the Source Listing display.

Indirect Command Examples

The following examples explain how to use indirect commands to correct the errors in the CA InterTest for CICS demo program COBDEMO.

1. To prevent the ASRA problem caused by the invalid decimal arithmetic data value assigned to the COBOL data-item TASKNUM:

   - Using Source Listing Option # 12 with a U line command, set an unconditional breakpoint at statement number 500.
   - Specify 10 as the indirect command statement number that you want to execute at the breakpoint.
   - Using the Source Listing Option # 11, access the indirect commands facility. Then, enter the indirect commands on the Indirect Commands screen, as shown next.

```
CA InterTest MONITOR COMMAND BUILDER    COBDEMO  INDIRECT COMMANDS     24

    Term ID (or .ANY or .NO): X001                  Delete ALL: NO

 LINE  COMMAND
-------------------------------------------------------------------------------
 00010  /* INITIALIZE TASKNUM */
 00020  MOVE 1 TO TASKNUM
 00030  EXIT


  1 Help       2            3 End        4           5               6
  7 Backward   8 Forward    9 Top       10 Bottom   11              12
  CAIN1261 Indirect commands added.
```

   Note: Setting TASKNUM to 1 alters the demo results. The clear key must be used to complete the termination of the demo when the "You have completed the sample CA InterTest for CICS test session" screen appears.

2. To avoid the storage violation that occurs in COBDEMO:

   - Using Source Listing option 12 with a U line command, set an unconditional breakpoint at statement number 978
   - Specify 50 as the indirect command statement number that you want to execute at the breakpoint

- Using the Source Listing Option # 11, access the indirect commands facility, and then add the following commands.

```
CA InterTest MONITOR COMMAND BUILDER    COBDEMO  INDIRECT COMMANDS 24
    Term ID (or .ANY or .NO): X001                  Delete ALL: NO
------------------------------------------------------------------------


 00010  /* INITIALIZE TASKNUM */
 00020  MOVE 1 TO TASKNUM
 00030  EXIT
 00040  *
 00050  /* CIRCUMVENT STORAGE VIOLATION */
 00060  GOTO 'SENDMAP00'

  1 Help       2            3 End        4           5               6
  7 Backward   8 Forward    9 Top       10 Bottom   11              12
  CAIN1261 Indirect commands added.
```

3. To perform the code that was bypassed at statement 1369 at another location where it will not cause a storage violation:

   - Using the Source Listing Option # 12 with a U line command, set an unconditional breakpoint at statement number 974
   - Specify 80 as the indirect command statement number that you want to execute at the breakpoint

- Using the Source Listing Option # 11, access the indirect commands facility, then add the following commands.

```
CA InterTest MONITOR COMMAND BUILDER    COBDEMO  INDIRECT COMMANDS    24

    Term ID (or .ANY or .NO): X001                  Delete ALL: NO
------------------------------------------------------------------------------

  LINE  COMMAND

 00010  /* INITIALIZE TASKNUM */
 00020  MOVE 0 TO TASKNUM
 00030  EXIT
 00040  *
 00050  /* CIRCUMVENT STORAGE VIOLATION */
 00060  GOTO 'SENDMAP00'
 00070  *
 00080  /* INSERT MOVE AT A SAFE LOCATION */
 00090  IF NEWDATA > ZEROES THEN MOVE NEWDATA TO STGAREA1
 00100  EXIT

  1 Help       2           3 End        4           5              6
  7 Backward   8 Forward    9 Top       10 Bottom   11             12
  CAIN1261 Indirect commands added.
```

4. To debug a program that is invoked by multiple transaction IDs and take breakpoints only when a particular transaction ID is specified:

- Using the Source Listing Option # 10 with a U line command, set an unconditional breakpoint at the beginning of the procedure division.
- Specify the indirect command statement number that you want to execute at the breakpoint.
- Specify the FROM Terminal, where the breakpoint should take effect, as .ANY.

  Now, all other program breakpoints that are set may be taken only when the conditions specified by the indirect command 10 are true.

```
CA InterTest MONITOR COMMAND BUILDER    COBDEMO  INDIRECT COMMANDS    24

    Term ID (or .ANY or .NO): X001                  Delete ALL: NO
------------------------------------------------------------------------------

  LINE  COMMAND

  00010 IF EIBTRNID =  'DEC2' THEN BREAK
  00020 RUN
  00030 EXIT

  1 Help       2           3 End        4           5              6
  7 Backward   8 Forward    9 Top       10 Bottom   11             12
  CAIN1261 Indirect commands added.
```

View Indirect Commands on the Status Report

The status report displays a list of all active indirect commands, and all active breakpoints and other monitoring options. Each line on the report shows the type of breakpoint, the presence of indirect commands, and the terminal at which they are set to take effect.

Access

To access the status report from the Source Listing display or breakpoint enter the command **STATUS**, or press **PF12** Status. The following screen shows how indirect commands are represented on the CA InterTest Status Report:

```
                      CA InterTest r8.5  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     COBOL
_ + | .ANY     User monitoring options   Active
_   | CARAR01  User monitoring options   Active
      |        Symbolic listing file     PROTSYM
_     | UBP    Unconditional breakpoint  'CONTINUE TASK'
      |        Option ID                 4B893C0
      |        From, to terminals        .ANY, .ANY
      |      ❶ Execute indirect command   100
_     | CMD  ❷ Indirect commands defined  .ANY
      |        Option ID                 90AB36F1
_     | SLB    Source listing breakpoints .ANY
         *** End of data ***



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

The previous figure shows the status report expanded for the program COBDEMO when monitored by user ID CARAR01. Notice the unconditional breakpoint entries and indirect command entries, which are discussed next.

❶

The UBP entry identifies the unconditional breakpoint set at the procedure-name Continue-Task. Notice the last line of the UBP description is Execute indirect command ... 100.

❷

The CMD entry identifies the indirect commands. Notice the indirect commands are assigned the option ID 90AB36F1.

To return to the Source Listing display from the Status Report, press **PF3** or **Clear**.

Edit Indirect Command Definitions

To view, add, change, or delete indirect commands directly from the Source Listing facility, enter **11** in the Option # field. CA InterTest for CICS displays the indirect commands defined for the default terminal, which is either the terminal you are working at or the terminal specified on the Source Listing Profile.

ITST Access to Indirect Commands

The ITST Monitoring Menus and the CNTL menu allow direct access to the indirect commands facility, where you can code or edit indirect commands to be executed by breakpoints.

From any menu or Source Listing screen, perform these steps:

1. Access the ITST Program Monitoring menu (ITST 2.1).

   The fast-path command entry from a breakpoint is:

   Command ==> **=1.2.1**.

2. On the Program Monitoring menu, complete the **Program name** and type **S** next to the Commands (indirect commands) Option. The entries are shown in the following screen.

```
     ------------CA InterTest r8.5  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                                          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)
s 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
```

3. Press Enter to continue to the next screen. The following Indirect Commands screen appears.

```
CA InterTest MONITORING COMMAND BUILDER  INDIRECT COMMANDS       24

    SET option to  EDIT  indirect commands defined for execution in:
                              PROG=COBDEMO

     Term ID (or .ANY or .NO) where commands will take effect:       ____
     user ID (or .ANY) who will execute the program:                 .ANY___



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

Optionally, specify the terminal ID where the indirect commands will take effect and the user ID that executes the program when the indirect commands are to take effect.

Note: Generally, accept the defaults. Press PF1 for help in completing these fields, if necessary.

To edit indirect commands previously coded for a program, ensure the terminal ID and user ID entries match those initially used to code the indirect commands.

Press **Enter** to continue. CA InterTest for CICS displays the Indirect Commands screen for the requested program, terminal ID, and user ID.

For details on how to insert insert, copy, edit, and delete the commands, see Format Indirect Commands (see page 204).

Indirect Command Syntax

This section discusses execution flow and syntax.

Execution Flow Control

If a breakpoint is set to invoke indirect commands, when you run your program the commands are automatically executed without stopping first. Execution is returned to the program according to the exi*t* **command** that you specified in your indirect commands.

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

BREAK

CA InterTest for CICS halts indirect commands processing and issues a breakpoint display from where the indirect commands were invoked in the program.

GOTO

CA InterTest for CICS resumes execution at the program statement number, offset, paragraph name, or indirect command statement number specified after the GOTO command.

EXIT

CA InterTest for CICS 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 for CICS resumes program execution at the breakpoint location from which the indirect commands were invoked and ignores all subsequent breakpoints.

Command Syntax

The syntax for the basic commands available in the indirect commands facility is described next.

Note: All bold characters are required in the syntax of the command.

Equate Command

This command equates a one- to eight-character symbol of your choice to a relatively long data name. A maximum of ten EQuate symbols can be active concurrently.

The syntax is:

EQuate *symbol* [TO] data-name

symbol

Indicates any user-defined symbol that observes the rules for data names.

data-name

Specifies the corresponding variable name.

Brackets, [ ], indicate an optional entry.

Example

```
EQ TN TASKNUMBER
```

Exit Command

The Exit command returns control to the statement following the last executed PErform indirect command. If no PErform is active, indirect processing is terminated and control is returned to the program, where execution continues from the breakpoint. Any relational operands entered with this command are ignored.

Example

```
EX
```

Goto Command

This command resumes execution at a specified location. The syntax controls where execution resumes.

To resume execution at the specified indirect command sequence number, use this syntax:

GOto *nnnnn

To resume execution at the specified statement number, use this syntax:

GOto #nnnnn

To resume execution at the specified hexadecimal displacement, use this syntax:

GOto +nnnnn

To resume execution at the specified paragraph-name, use this syntax:

GOto '*paragraph-name*'

Example

```
GO *1150
```

IF, THEN Statement

This statement tests a conditional expression. If the expression is true, the THEN portion is processed.

The syntax is:

IF expression **THEN** indirect command

expression

Specifies any valid condition.

indirect command

Specifies any valid indirect command.

Example

```
IF (A+B)*C=D
  THEN GOTO *1150
```

Break and Pause Commands

The BReak and PAuse commands terminate indirect command processing and redisplay the Breakpoint display. Any operands included with these commands are ignored by CA InterTest for CICS.

Example

```
BR
```

Move Command

This command moves the specified source-item to the target data-name. Follow COBOL MOVE statement rules when formatting an indirect command Move.

The syntax is:

```
Move [source-item] TO data-name
```

Note: Brackets,[ ], indicate a choice of source-items.

Valid source-items follow.

Source Item	Description
[ALL,]'*char-data*' [ALL,]"*char-data*"	*'char-data'* or "*char-data*" is a non-numeric character string.
[ALL,]*x*'*hex-data*' [ALL,]*x*"*hex-data*"	*x'hex-data'* or *x*"*hex-data*" is a hexadecimal string.
numeric literal	*numeric literal* is valid only if the target data-name is a numeric variable.
figurative constant	*figurative constant* is any valid COBOL figurative constant.
floating-point literal	*floating-point literal* is valid only if the target data-name is COMPUTATIONAL-1, COMPUTATIONAL-2 (COBOL) or FLOAT (PL/I).
arithmetic expression	*arithmetic expression* is any valid CA InterTest arithmetic expression.
bit literal	*bit literal* is any bit literal string.

Note: For a complete description of each of these source items, see Formatting Indirect Commands later in this chapter.

data-item

Refers to a COBOL or PL/I variable name. Qualifiers and subscripting are allowed.

data-name

Refers to another COBOL or PL/I variable name. If the data-name is a COBOL Index cell, the MOVE command moves exactly the data specified into the Index cell. No mathematical calculation is performed.

Perform Command

This command transfers control to the specified indirect command sequence number as a subroutine call. A maximum of five PErform commands can be active concurrently.

The syntax is:

PE`rform` *nnnnn*

nnnnn

Indicates the indirect command sequence number to which control is transferred.

Example

PE 2250

Format Indirect Commands

Rules

Indirect commands and their arguments are free-form. However, there are several rules to keep in mind as you format your commands:

- All arguments must be separated by at least one blank space.
- Only one indirect command displays on a line (with the exception of the IF command), but one indirect command spans several lines.
  - The first line of a command must begin with a line number followed by a blank space
  - Consecutive lines of a single command must begin with at least one blank space

Example

```
10  IF DATA-COUNTER <> 0
    THEN M 0 TO DATA-COUNTER
20  BR
```

- New indirect commands are entered by:
  - Overtyping an existing command
  - Entering new indirect commands on blank lines below existing ones

Example

If you have indirect command 10 followed by indirect command 20, to insert commands between the two you would tab to the next blank line, enter indirect commands 11 through 19. Press Enter. CA InterTest for CICS places the commands in numeric sequence.

- Duplicate indirect commands by changing only the sequence number of an existing indirect command. The original command is not affected; it is duplicated at the specified sequence number.

  Note: Be sure to overtype the entire line number when duplicating a command. Failure to do so can result in an erroneous move. For example, if you want command 00010 to be copied to line 00045, you must overtype the entire line number 00010 with 00045 or 45 b b b (where b is a blank). If you only overtype the first two digits with 45, the command will be copied to line 45010, not 00045.

- Enter comments in place of commands. To enter a comment, specify a sequence number followed by at least one space, and then enter either an asterisk (*) or a backslash and an asterisk (/*) followed by at least one space and your comment.

Example:

```
00035  * THIS IS A COMMENT LINE
```

- Indirect commands are verified after you press Enter, but some errors might not be detected until the indirect commands are actually processed.
- Delete an entire set of indirect commands by specifying YES in the Delete ALL= field.

Delete a specific indirect command by overtyping the line number with DEL.

Data Names and Variables

Indirect commands reference any data name in the Data Division Linkage section of the program that you are testing.

For COBOL (prior to COBOL II Release 1.3)

- One group-item qualifier is allowed. It is any group that contains the item in question:
  - Immediate parent
  - 01-level
  - Any level in between the immediate parent and the 01-level
- Up to three levels of subscripting are allowed. Subscripts include:
  - Positive integer numeric literal
  - Another variable defined as a positive integer numeric field (no sign or decimal portion)
  - INDEX item (either INDEXED BY or USAGE INDEX)

For COBOL II (Releases 1.3 and higher) and COBOL/370

- One program-ID qualifier is allowed.

  Note: If your COBOL II source listing contains a nested program that has a local variable with a name identical to a variable in the main program, use the appropriate program-ID to indicate which variable you are referencing.

- One group-item qualifier is allowed. It can be any group containing the item in question:
  - Immediate parent
  - 01-level
  - Any level in between the immediate parent and the 01-level
- Up to seven levels of subscripting are allowed. Subscripts include:
  - Positive integer numeric literal
  - Another variable defined as a positive integer numeric field (no sign or decimal portion)
  - INDEX item (either INDEXED BY or USAGE INDEX)

Example

```
DATE OF INVOICE
TABLE-ENTRY(4,TABLE-INDEX)
```

For PL/I Programs

- Only items declared as STATIC, AUTOMATIC, BASED, and POINTER type variables that are declared as parameters to the entry point of the MAIN procedure are supported.
- One procedure-name qualifier is allowed. If your PL/I source contains an internal procedure that has a local variable with a name matching a variable in the main procedure, use the appropriate procedure-name to indicate to which variable you are referencing.
- Up to 15 structure-name qualifiers are allowed. You must follow PL/I notation conventions; for example: A.B.C.D.E.
- Up to 15 levels of subscripting are allowed. A subscript includes:
  - Integer numeric literal
  - Another variable defined as an integer numeric field (no decimal portion)
- The subscript value must be within the actual declared bounds of the array.

Example

If the array is declared as (100:200), subscript values must be within the range of 100 to 200, inclusive.

- The base address for a variable is supplied with the base locator notation. One item supplies the base address for the next item, which in turn supplies the base address for the next variable, up to a limit of 15 items. This feature is intended for BASED variables, but it is used to map any declaration over any addressable storage.

Example:

```
DATE OF INVOICE
TABLE_ENTRY(4,TABLE_INDEX)
```

Literals in Indirect Commands

Numeric is a string of one to eighteen digits with no intervening commands or decimal point, such as 7954. If the target variable contains an implied decimal point, then the decimal point must also be implied in the literal.

Floating point is appropriate only for COMPUTATIONAL-1 and COMPUTATIONAL-2 target variables that consist of the following:

- A sign
- A string of up to seven digits with an optional decimal point
- The character E followed by an optional signed exponent

Alphanumeric is a character string enclosed within single quotation marks ('), or double quotation marks ("), such as 'RALPH9' or "RALPH9".

Hexadecimal is a string of an even number of characters consisting of 0 to 9 and A to F, enclosed within single quotation marks ('), or double quotation marks ("), and prefixed with *X*, such as X'F0'.

Bit is a string of the characters 1 and 0 enclosed in single quotation marks ('), or double quotation marks (") such as '11110000'. The maximum length of a bit literal is determined by its context and the maximum allowed indirect command length.

Figurative Constants in Indirect Commands

Use the following figurative constants in formatting indirect commands:

Valid Figurative Constants			
ZERO ZEROS ZEROES	numeric value 0	HIGH-VALUE HIGH-VALUES	character value X'FF'
SPACE	blank (X'40')	LOW-VALUE	character value X'00'

Valid Figurative Constants			
SPACES		LOW-VALUES	
QUOTE QUOTES	quotation mark (')	ALL *literal*	any *literal* to fill a field

Conditional Expressions in Indirect Commands

Indirect commands support the following types of expressions:

- Conditional expressions, used with indirect commands IF statements
- Arithmetic expressions, used with indirect commands IF and MOVE statements

Operands in these expressions are one of the following expressions:

- Figurative constants
- Literals
- Variables

  Note: Qualified data names and subscripts are also supported.

The following is a list of acceptable relational operators.

Relational Operator	Meaning
=	equal to
< >	not equal to
<	less than
>	greater than
< =	less than or equal to
> =	greater than or equal to

COBOL Variable Syntax

The general syntax for a COBOL variable in CA InterTest for CICS indirect commands is as follows:

```
[prog-id:] data-name [IN grp-name][ ({sub-name OR ind-name OR number}[, ...]) ]
```

or

```
[prog-id:] data-name [OF grp-name][ ({sub-name OR ind-name OR number}[, ...]) ]
```

prog-id

Identifies the program-ID of the program to which the variable belongs. If used, the *prog-id* must immediately be followed by a colon (:). If omitted, the default is the main program's program-ID.

data-name

Indicates COBOL data item.

grp-name

Identifies the COBOL identifier of a group item to which the data-name belongs. This clause lets qualification of an identifier that displays in more than one group.

sub-name

Specifies a COBOL identifier to use as a subscript. This identifier must follow all the COBOL rules for valid subscripts.

ind-name

Indicates a COBOL index-name.

number

Specifies a positive integer that is used as a valid subscript for a data-name.

Note: OR is not part of the command syntax.

Examples:

The following are examples of valid COBOL data-names:

```
TOTAL-AMT
CUST-NO OF CUST-REC
MONTHLY-BAL (8)
YEARLY-TOT (W-YR, 3, W-DEPT)
 SUBRTN1:ITEM-A
SUBRTN1:ITEM-B IN INP-REC(3, SUB-2, 4, SUB-3)
```

PL/I Variable Syntax

The general syntax for a PL/I variable in CA InterTest for CICS indirect commands is as follows:

```
expression-1 [-> expression-2 ...]
```

->

Defines **the address for** symbol. It indicates that the expression on the left provides the address on which the the expression on the right is based. If the expression on the left is a PL/I POINTER variable, the contents of the POINTER variable is use. The -> notation is provided for referencing PL/I BASED variables whose addresses cannot be resolved at compile time, but are used for referencing any PL/I variable.

expression is defined as follows:

```
[procnm:] var-name-1[,var-name-2 ...] [({sub-name OR number}[, ...])]
```

procnm

Indicates the label of the procedure to which the variable belongs. If used, *procnm* must immediately be followed by a colon (:). If omitted, it defaults to the MAIN procedure name.

var-name

Identifies the PL/I variable name.

sub-name

Defines a PL/I variable name to use as a subscript. This identifier must follow all PL/I rules for valid subscripts.

number

Specifies a positive integer that is a valid subscript for *var-name*.

Note: OR is not part of the command syntax.

Examples

The following are examples of valid PL/I variables:

```
TOTALAMT
CUSTREC.CUSTNO.PREFIX
MONTHLYBAL (8)
 YEARLYTOT (WYR, 3, WDEPT)
 SUBRTN1:ITEMA
SUBRTN1:INPREC.ITEMB (3, SUB2, 4, SUB3)
```

Example

PRT1 is a POINTER variable that provides the base address for BASED variable BASEDSTRUCT1.

```
PRT1 -> BASEDSTRUCT1
```

Example

PRT1 is a POINTER variable that provides the base address for BASED variable BASEDSTRUCT1. It contains another POINTER variable, PRT2, which is used as the base address for the BASED variable BASEDSTRUCT2, containing ITEMA.

```
PRT1 -> BASEDSTRUCT1.PRT2 -> BASEDSTRUCT2.ITEMA
```

Example

STRUCTA is not a pointer variable but a structure. The address of STRUCTA is used to map the variable names defined in STRUCTB.

```
STRUCTA -> STRUCTB
```

Chapter 6: Resume and Execution Options

This chapter discusses how to control and resume execution from a breakpoint.

This section contains the following topics:

Control and Resume Execution from a Breakpoint (see page 213)
The Resume Menu (see page 218)
Abend the Task (see page 222)
Review Program Status After Execution (see page 223)
Your Program Source Code (see page 223)
Continue to Test (see page 225)

Control and Resume Execution from a Breakpoint

You can control program execution from a breakpoint in several ways:

- Single-stepping through the execution
- Setting additional breakpoints
- Jumping to another location
- Resume program execution ignoring all breakpoints

Single-step Through Execution

An effective way to debug a program is to reach a breakpoint, add data items whose values you want to track to the Keep window, and execute a few instructions at a time or single-step through the program to see how the values change.

Set Additional Breakpoints

You also can set additional breakpoints in your program. This means that you do not have to decide in advance which program paths to execute for the entire test session. Decide at any breakpoint how to continue your test session.

Jump to Another Location

At a breakpoint, continue execution from where the program stopped or from another location to:

- Go around a problem area
- Force execution of bypassed code
- Change the logical flow for testing purposes

Continue Execution

To continue execution from an automatic breakpoint, you must do one of the following:

- Dynamically correct the error. To dynamically modify main storage to correct the error and for instructions on using the indirect commands facility.
- Set a CA InterTest for CICS protection option, discussed in the chapter "Monitoring Menu Options."
- Go around the error. For details, Jump to Another Location (see page 214) and Using the Resume Menu (see page 218). Options 2, 3, and 4 on the Resume menu offer easy ways to specify where you want to continue execution.

CA InterTest for CICS lets you control where execution resumes and how much code executes at a time. Resume execution from where you stopped or from another location; and either single-step through execution or continue until CA InterTest for CICS reaches the next breakpoint or the end of the program.

Press **PF5** or specify **GO** on the command line to continue from the current breakpoint. Execution continues until another breakpoint occurs or the program ends, whichever comes first.

For more information:

Breakpoint Activities (see page 135)
Monitoring Menu Options (see page 227)

Single-Step Through Execution

Press **PF10** or enter the **NEXT** command on the command line to single-step through a specific number of verbs, instructions, or statements. The default is one verb, instruction or statement, but this can be changed by specifying **NEXT value** or pressing PF4 to access the Source Listing Profile screen.

When stopped on a PERFORM or CALL, the **NEXT OVER** command causes the test session to continue until the statement following the PERFORM or CALL is encountered, where a break point occurs.

When stopped within a paragraph or subroutine, the **NEXT RETURN** command causes the test session to continue until the session encounters the statement following the PERFORM or CALL that invoked the current paragraph or subroutine, where a break point occurs. You must have TRACE ON to use **NEXT RETURN.**

Note: When you use **PF10** or the **NEXT** command to single-step through nested COBOL IF statements, it might appear that CA InterTest for CICS is not executing one verb at a time. This is a function of how the compiler generates the object code.

Important! To more closely follow the execution of nested IF statements, use the **PF9** on the Detailed Breakpoint display to execute one instruction at a time.

For more information

Source Listing Facility (see page 47)
Breakpoint Activities (see page 135)

Jump to Another Location

To specify a new location for resuming execution, follow these steps:

1. Specify the **GO line-number|label** command or:
   a. Display the statement or instruction where you want to resume execution.
   b. Enter **g** to the left of the line number in column 1.
   c. Press **PF5** to continue to the next breakpoint or until program execution ends, or press **PF10** to single-step execution.

Example

The following screen shows how to resume execution from a statement.

```
             CA InterTest  PROTDEM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt  4 Profile    5 Resume     6
     7 Backward  8 Forward   9           10 001 Verb  11 Backtrace 12 Status
------------------------------------------------------------------------------
    g 01279 READDATASET.
      01280*    EXEC CICS HANDLE CONDITION
      01281*              DSIDERR
      01282*              NOTOPEN(NOTOPEN)
      01283*              ENDEXEC.
    _ 01284      MOVE '                  ' TO DFHEIV0 CALL 'DFHEI1' USING
    _ 01285      DFHEIV0 GO TO NOTOPEN DEPENDING ON DFHEIGDI.
      01286
      01287
    _ 01288     MOVE ZEROES TO RECORDKEY.
      01289**** NOTE * DATASET NAME IS WRONG ****
      01290*    EXEC CICS STARTBR
      01291*              DATASET('PROTH')
      01292*              RIDFLD(RECORDKEY)
      01293*              ENDEXEC.
    A   ==>      MOVE 'PROTH' TO DFHEIV3 MOVE '         ' TO DFHEIV0 CALL
```

Notes:

- Use this option with caution in COBOL and Assembler programs and optimized code. Certain uses trigger difficulties in getting proper addressability.
- When an error occurs in a paragraph containing a CICS command, resume program execution at the paragraph-name rather than at the specific instruction, because some CICS internal processing might have occurred prior to that command.
- In PL/I programs, do not resume execution at a new location if it results in a jump between PROCS/BLOCKS.

For more information:

CNTL Commands and Menus (see page 295)

Debug Optimized Applications

CA InterTest for CICS supports the debugging of programs that have been optimized, either by the COBOL compiler's OPTIMIZE option or by CA Optimizer or CA Optimizer/II. However, debugging these programs sometimes results in unexpected behavior.

Note: The PL/I compilers are optimizing compilers.

Often, as part of the optimization process, a compiler will relocate individual instructions, statements, or even entire paragraphs so that the optimized program will run more efficiently. This means that some or all of the instructions generated for a given statement may be moved to another statement, or that some or all of the statements in a paragraph may be moved to another paragraph. When this type of optimization occurs, the resulting object program and corresponding listing may not accurately represent the relationship between the source statements and their generated object code, or even between a paragraph label and the statements contained within the paragraph. As a result, there may be times when the breakpoint intercept does not occur, or when the wrong sequence of statements appears to be executed while single-stepping. There may also be times when the debugger appears to highlight the wrong statement at a breakpoint intercept.

These unexpected displays do not indicate that a program is being executed incorrectly. They indicate that the debugger sometimes cannot accurately identify which object code corresponds to which source statement, or which statement is contained within which paragraph.

CA InterTest for CICS uses the information in the compiler-generated procedure map or offset report to establish the program offset for each statement and label in the program. During execution, the debugger recognizes the start of the new statement or label by matching the program offset of the currently executing instruction with the PROTSYM information obtained from the compiler listing. Therefore, the accuracy with which the debugger represents a breakpoint or other intercept is only as good as the information in the compiler listing.

Inaccuracies may include, but will not be limited to:

- Incorrect execution when using GO line-number.
- Failure to stop at a breakpoint at a paragraph label or statement.
- Unexpected or out-of-sequence highlighting of statements when single-stepping.

Additionally, application abends may result from the use of the GO line-number command because the optimized object code may have register requirements that do not support changes to the flow of control. These commands should be avoided when debugging an optimized program.

For the best debugging results, avoid using optimization whenever possible in your testing environment. Production applications may be compiled with optimization, and debugging these applications as they exist without recompiling is supported. However, you may experience some of the inaccuracies listed previously under these circumstances.

Resume Program Execution Ignoring All Breakpoints

To continue processing from the current statement ignoring all preset breakpoints and indirect commands until the task either abends or runs to normal completion, specify the **RUN** command on the Source Listing Breakpoint Command Line.

The Resume Menu

Option 4 Resume on the Breakpoint Primary Option menu displays the Resume menu, shown in the following illustration. The fast-path command entry to access the Resume menu from any breakpoint display is:

```
COMMAND ===> =4
```

```
 -------CA InterTest r8.5  RESUME BREAKPOINTED TASK MENU--------
OPTION  ===>

   1  Next instruction     Resume from the next sequential instruction
   2  Label                Resume from a label
   3  Statement            Resume from a statement number
   4  Offset               Resume from an offset
   5  Indirect command     Execute an indirect command sequence
   6  Next EXEC/CALL       Resume until the next EXEC command or external call
   7  nnn steps            Resume for specified number of statements or verbs
   8  nnn instructions     Resume for specified number of machine instructions


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

Resume at the Next Instruction

Resume Menu Option **1 Next** Instruction immediately resumes execution at the next instruction. This is the same as using PF5 Resume on the Breakpoint display. The fast-path entry from the Breakpoint display is **=4.1**.

Resume Task from a Label

Resume Menu Option **2 Label** displays the following menu where you enter the name of a label from which to resume execution. The fast-path entry to display this menu from a breakpoint is **=4.2**.

```
-----------  CA InterTest r8.5  RESUME TASK FROM A LABEL ---------
 COMMAND ===>

    Type the label name, then press ENTER.


    Label  _________________________________   No quotes
```

After entering a label name, press **Enter** to continue. The program resumes execution from the label you specified.

Resume Task from a Statement

Resume Menu Option **3 Statement** displays the following menu where you enter a statement number from which to resume execution. The fast-path entry to display this menu from a breakpoint is **=4.3**.

```
-------------  CA InterTest r8.5  RESUME TASK FROM A STATEMENT NUMBER ---------
 COMMAND ===>

    Type the statement number, then press ENTER.


    Statement number  _____
```

After entering a statement number, press **Enter** to continue. The program resumes execution from the label you specified.

Resume Task from an Offset

Resume Menu Option **4 Offset** displays the following menu, from which you enter an offset from which to resume execution. The fast-path entry to display this menu from a breakpoint **is =4.4**.

```
--------------  CA InterTest r8.5  RESUME TASK FROM AN OFFSET -------------
 COMMAND ===>

    Type the offset, then press ENTER.


    Offset  _____   Hexadecimal value
```

After entering the hexadecimal value of an offset, press **Enter** to continue. The program resumes execution from the offset you specified.

Resume Task from an Indirect Command Sequence

The Resume Menu Option **5 Indirect Commands** lets you specify the statement number of an indirect command. CA InterTest for CICS immediately continues execution with that indirect command. The fast-path entry to display this menu from a breakpoint is **=4.5**.

```
---------- CA InterTest r8.5  EXECUTE AN INDIRECT COMMAND SEQUENCE ------------
 COMMAND ===>

    Type the sequence number, then press ENTER.


    Sequence number  _____
```

For more information:

Breakpoint Activities (see page 135)

Resume Until Next EXEC/CALL

The Resume menu option **6 Next EXEC/CALL** lets you execute until the next EXEC command or external call in your program. For example, using this option might produce the breakpoint shown in the following illustration, which is stopped at an external call. The fast-path entry to display this menu from a breakpoint is **=4.6**.

```
              CA InterTest  PROTSYM FILE SOURCE LISTING BREAKPOINT
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 Det Bkpt  4 Profile    5 Resume     6 Menu
     7 Backward  8 Forward   9           10 001 Verb  11 Backtrace 12 Status
--------------------------------------------------------------------------------
-
   00912*          ENDEXEC.
 _ 00913      MOVE '  Y      00624   ' TO DFHEIV0
 _   ==>     CALL 'DFHEI1' USING DFHEIV0  TSQNAME TASKSTRUCTURE TSQLEN
 _ 00915     DFHDUMMY TSQITEM.
   00916
   00917
   00918
.
.
.
```

Resume Task for nnn Steps

Resume Menu Option **7 nnn Steps** gives you a menu to enter the number of steps you want to execute at this time. The fast-path entry to display this menu from a breakpoint is **=4.7**.

```
------------ CA InterTest r8.5  RESUME FOR NNN STEPS-----------------
 COMMAND ===>

    Type the number of steps, then press ENTER.


    Steps  _____   (132767), definition differs by program language:

                   COBOL       number of verbs
                   PL/I        number of statements
                   ASSEMBLER   number of statements
```

This option is a one-time execution of the steps you specify and does not change the value of PF 10 on a Breakpoint display. To set the value for PF 10, use the Source Listing Profile, PF4 from any Source Listing display.

For more information:

Source Listing Facility (see page 47)

Resume Task for nnn Machine Instructions

Resume Menu Option **8 nnn Machine Instructions** gives you a menu to enter the number of instructions you want to execute at this time. The fast-path entry to display this menu from a breakpoint is **=4.8**.

```
-------  CA InterTest r8.5  RESUME FOR NNN MACHINE INSTRUCTIONS---------------
 COMMAND ===>

    Type the number of machine instructions, then press ENTER.


    Machine instructions  _____   (132767)
```

This option is a one-time execution of the steps you specify and does not change the value of PF9 on a Detailed Breakpoint Display.

Abend the Task

You have the option to abend a task, with or without a dump, at any breakpoint. To abend a task from a breakpoint, follow these steps:

1. Restore the breakpoint display by pressing **Clear**. If you are returning from the CA InterTest for CICS CNTL, CORE, HELP or FILE facilities, you might need to press **Clear** repeatedly.
2. Enter **ABEND** on the command line and press Enter. CA InterTest for CICS prompts you to enter an Abend code if you want a dump, as shown next.

```
----------- CA InterTest r8.5  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
.
.
.
```

3. From the Abend menu screen, you have several options:
   - Press **Clear** to cancel the abend and return to the breakpoint display.
   - To abend **without a dump**:
     - To take a normal abend without a dump, accept an abend code of blanks and press Enter.
     - To cancel all program exits without a dump, specify the code: *XXXX*, and press Enter.
   - To abend **with a dump**, specify any four-character dump code **except** *XXXX*. Do not specify a code starting with the letter A.

CA InterTest for CICS displays a message confirming the abend request and dump code. Follow any online instructions to complete the abend. After an abend, you return to CICS.

Review Program Status After Execution

When your transaction terminates normally or with an abend, you return to CICS. At this point, review the following items:

- Status of your program source code
- CA InterTest for CICS monitoring and monitoring options
- Auxiliary files and databases accessed by your program

Your Program Source Code

Your program source code is not modified. It is exactly the same as before you began testing.

Example

If CA InterTest for CICS detected an ASRA and you used CA InterTest for CICS to correct the error, the original error still exists in your code.

If you modified main storage, that change was only temporary—**your program remains unchanged**.

The CA InterTest for CICS copy of your program from the compiled or assembled output also remains unchanged. If you rerun the transaction, you will see the same display of your program in the Source Listing Facility as you did the first time.

CA InterTest for CICS Monitoring and Monitoring Options

CA InterTest for CICS monitoring and monitoring options, which you set before or during testing, remain in effect until you remove them. These include all breakpoints set before or during program execution that were not removed during the test session.

To get a status display of monitoring activity and options in effect for one program, perform these steps:

1. Display the Source for the program.
2. Enter the command **STATUS**.

To get a status display of monitoring activity and options in effect for all programs, enter the command **STATUS ALL**.

To get global status display from the Primary Option Menu, perform these steps:

1. Access the Primary Option Menu.
2. Select Option **2 Monitoring**.
3. Select Option **4 Status**.

Note: The fast-path entry for a Monitoring Status Report from CICS is **ITST 2.4**

Alternatively, enter one of the following commands on a clear screen under CICS:

- For a program:

  CNTL=INQ,PROG=progname

- For multiple programs:

  CNTL=INQ,PROG=(prog1,prog2,prog3)

Example

To display the CA InterTest for CICS monitoring status for the program COBDEMO, enter the following command under CICS:

```
CNTL=INQ,PROG=COBDEMO
```

Files and Databases

Any files or databases updated by the program are permanently modified unless the No File Updating (NUP) option was set in advance. The NUP option is set using any Monitoring Menu.

Important! File or database changes that you make using the CA InterTest for CICS FILE facility are permanent, whether or not the NUP option is set.

For more information:

Monitoring Menu Options (see page 227)
CNTL Commands and Menus (see page 295)

Continue to Test

In some testing situations, you might want to re-execute the program without changing the source code. Doing this means you will see the same automatic breakpoints that you saw the first time. When it is easy to correct or get around the errors, this method can be the most efficient route to continue testing because you do not have to recompile your program.

Use the indirect commands facility to continue to test your program without recompiling it.

For more information:

Breakpoint Activities (see page 135)

Correct Your Source Code

When you decide to continue testing from a new version of your program, perform these steps:

1. Modify the program source code (outside CA InterTest for CICS).
2. Recompile or re-assemble the modified programs using the CA InterTest for CICS post-processor, as discussed in the chapter "Source Listing Facility." The CA InterTest for CICS post-processor automatically updates the CA InterTest for CICS Symbolic File and the Source Listing facility with the new version of the program.
3. Execute the CA InterTest for CICS New Program Copy option, if desired. This option is on the Program Monitoring menu (2.1). New Copy performs the following functions:
   - Resets the CICS entry in the PPT table to the program's new library address
   - Transfers all symbolically set breakpoints and other monitoring options to the recompiled program

Note: To recompile your program and keep the same breakpoints in the recompiled version, set breakpoints at **paragraph names** or **labels**, not at lines of executable code. Then use the CA InterTest for CICS New Program Copy function to transfer the breakpoints to the recompiled program. If you set breakpoints at lines of executable code, New Program Copy transfers the breakpoints to the same **statement numbers** or **offsets** in the recompiled program, which might not be what you want.

For more information:

Monitoring Menu Options (see page 227)

Reset Breakpoints after Recompiling or Re-Assembling

If breakpoints were set at executable statements or locations, you might need to adjust their locations after recompiling a program, as follows:

- To reset the breakpoint locations, adjust the locations one at a time or remove them all and re-enter them.
- To scroll through your program one breakpoint at a time, set the scroll amount to STOP on the Source Listing Profile screen, accessed by typing **PROFILE** or pressing **PF4** from a Source Listing display.
- To remove all breakpoints at once for a program that is being monitored with User ID=.ANY, you must first check that no one else is executing the program before you remove monitoring. Get a Monitoring Status Display for the program. If no one else is executing the program, remove monitoring or remove all breakpoints directly from the Monitoring Status report.

Important! Use this option with caution! Removing monitoring for a program clears all breakpoints and all monitoring options.

Chapter 7: Monitoring Menu Options

The primary task of CA InterTest for CICS is to monitor programs to detect and prevent errors. When you set monitoring for a program, CA InterTest for CICS detects and prevents errors before they occur. That means monitored programs will not abend or commit storage violations.

This chapter describes the Monitoring Menu Options (Option 2 on the Primary Option Menu) that let you take advantage of the full range of CA InterTest for CICS monitoring capabilities. For example, use the monitoring menus to set monitoring for programs, all programs executed by a transaction, or all the programs executed from a terminal. Use the menu to invoke many options for monitoring, such as breakpoints, data monitoring, indirect commands, CICS resource replacement options, and system-wide options. The monitoring menus also enable you to get a selective Monitoring Status report.

You might also be interested in these related topics.

Monitor Status

For more information on how to access and use the Primary Option Menu and Status Report, see the chapter Menus and Displays (see page 35).

Breakpoints

For more information on how to set or remove breakpoints from the Monitoring Menu, see the chapter Using Breakpoints (see page 103).

Data Monitoring

For more information on how to use Data Monitoring, see the chapter Breakpoint Activities (see page 135).

Indirect Commands

For more information on how to use indirect commands, see the chapter Breakpoint Activities (see page 135).

CNTL Commands

To use the CNTL transaction, alternate menus, and transaction-based commands to specify monitoring options.

For more information:

CNTL Commands and Menus (see page 295)

Monitoring Menu Functions (ITST 2)

The Monitoring Menu has the following options to perform monitoring-related functions:

- Programs
- Transactions
- Terminals
- Status
- Active tasks
- System-wide

Programs

Displays the Program Monitoring screen, which lets you set and remove monitoring, breakpoints, monitoring options, and monitoring-related functions for one or more programs. Monitoring related functions include the following items:

- Status report to display and remove monitoring and monitoring options for programs, transactions, and terminals
- Setting and removing monitoring, breakpoints, and monitoring options
- Setting and removing statement tracing and data monitoring
- New copy function to reset monitoring for a recompiled program
- Indirect commands to enable inline adjustments to programs without recompiling
- Composite support to provide symbolic support for separately compiled programs brought together when they are link-edited

Transactions

Displays the Transaction Monitoring screen which lets you set and remove monitoring, and monitoring options for one or more transactions. Monitoring options for transactions include request breakpoints, statement tracing and data monitoring, and replace, protect and special options.

Terminals

Displays the Terminal Monitoring screen, which lets you set and remove monitoring and monitoring options for all the programs executing from a terminal.

Status

Displays the status report (2.4) for all monitoring activity in the current CICS region. Use the status report to quickly remove a monitoring option or monitoring entry.

Active Tasks

Displays the list of active tasks that are being monitored by CA InterTest. If the system-wide purge option is turned on, the Active Tasks screen lets you to purge a task. If you previously disconnected a task from a terminal, the Active Tasks screen lets you reconnect the task and continue testing.

System-Wide

Displays the System-Wide Options submenu (2.6), which lets you display and modify monitoring options for the entire system. These include automatic breakpointing, checkpointing, dumping all abends, global logging, CA InterTest for CICS tracing, and breakpointed task purging.

Note: Additional monitoring options, specified by CNTL commands, are available to perform more specialized functions.

For more information:

CNTL Commands and Menus (see page 295)

Access

The Monitoring Menu is Option 2 on the Primary Option Menu.

To access the Monitoring Menu from a Source Listing display, perform these steps:

1. Type the command **MENU** or press **PF6** to access the Primary Option Menu.
2. Select Option **2**.

Note: The fast-path command entry is **=2.**

To access the Monitoring Menu from a Source Listing Breakpoint display, type the command **MENU** or press **PF6** to access the Breakpoint Primary Option Menu, and then select Option **1.2**. The fast-path command entry is =1.2.

PF Keys

Exit any Monitoring Menu by pressing a PF key. The following table describes the function of each PF key.

PF Key	Function
PF1 or 13	Initiates the CA InterTest for CICS online help facility.
PF2 or 14	Unassigned
PF3 or 15	Terminates the current processing and returns to the previous display.
PF4 or 16	Returns you to the top-level menu, the Primary Option Menu.
PF5 or 17	Unassigned
PF6 or 18	Unassigned
PF7	Scrolls a display list backward towards the top of the list.
PF8	Scrolls a display list forward towards the bottom of the list.
PF9 to 12 PF21 to 24	Unassigned

The User ID Monitoring Option

When you set monitoring, breakpoints or options for a specific program, transaction, or terminal, CA InterTest for CICS creates an entry for that program, transaction, or terminal in the monitoring table. For releases after Version 5.3, each monitoring table entry also includes a user monitoring option. The user option is .ANY or a specific CICS user ID. The user option indicates for whom the program, transaction, or terminal will be monitored. The following table describes how the value of the user monitoring option affects monitoring and breakpoints.

User Monitoring Option	Description
User ID = .ANY	Everyone is monitored, regardless of the CICS user ID. Breakpoints are directed to the terminals where they were set unless a specific option was set to redirect the breakpoints.
User ID = *specific CICS user ID*	Monitoring occurs only for the specified user based on the CICS user ID. This restricts the monitoring entry to a single user in a secure CICS system, and lets you move from one terminal to another, preserving breakpoint settings and options.

User ID Set to .ANY

The following monitoring status report shows that program COBDEMO will be monitored for ANY user of the program. The unconditional breakpoint (UBP) displays on the Source Listing Breakpoint screen on the same terminal where it was set, terminal ID X508.

```
  ---------CA InterTest r8.5   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     COBOL
_   | .ANY     User monitoring options   Active
    | |        Symbolic listing file     PROTSYM
_   | | UBP    Unconditional breakpoint  #1
    | |        Option ID                 DD103243
    | |        From, to terminals        X508, X508
_   | | SLB    Source listing breakpoints X508
```

User ID Set to Specific

The following status report shows the same program COBDEMO, which will be monitored for a single user, MARY01. The unconditional breakpoint is directed to any terminal where CICS user MARY01 is signed on.

```
   ------CA InterTest r8.5   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     COBOL
_   | MARY01   User monitoring options   Active
      |        Symbolic listing file     PROTSYM
_     | UBP    Unconditional breakpoint  #1
      |        Option ID                 DD103243
      |        From, to terminals        .ANY, .ANY
_     | SLB    Source listing breakpoints .ANY
```

Monitoring with a specific user ID gives you the ability to do the following actions:

- Isolate monitoring and debugging to a specific user, and not all users
- Move from terminal to terminal (or use a multi-session manager) and automatically redirect monitoring options and breakpoints to your current session, provided the from and to terminals default to .ANY
- Have two users concurrently debug a program, but with different monitoring options

What Is Your User ID Default?

The default user ID appears in the User ID field on the Source Listing Profile display. On the submenus related to Monitoring (Program Monitoring, Transaction Monitoring, and Terminal Monitoring), issue the USER command.

- If you are working in a CICS region without login security, the user ID value is always set to .ANY
- If you are working in a secure CICS region, the CA InterTest for CICS installation option DFLTUSER determines whether the default value assigned to the user ID is .ANY or your CICS user ID

The following table explains how the default values work when you are and are not signed on to CICS.

DFLTUSER Setting	Signed on to CICS	Not Signed on to CICS
.ANY	By default, everyone is monitored. Breakpoints are directed to the terminal where they were set.	Everyone is monitored. Breakpoints are directed to the terminal where they were set.
specific	By default, monitoring applies only to your user ID and breakpoints are displayed at whatever terminal you are using.	Not recommended, but takes the value of the default CICS user ID specified by the Systems Programmer in the CICS startup JCL. Type USER command to display this value.

Look up the current DFLTUSER value from the Primary Option Menu (fast-path 7.1.2) by performing these steps:

1. Use option 7 Status/Maintenance.
2. Select Option 1 Product Status.

3. Select Option 2 Global Options

This displays all installation options for IN25OPTS in alphabetic order. The DFLTUSER setting shows the default user ID setting for your system.

Override the User ID Default

Override the default user ID setting using any of the following methods:

- From the Source Listing Facility, change the user ID on the Profile screen **before** you set monitoring or breakpoints from the Source Listing Display
- From the Monitoring menus, change the user ID on the Program Monitoring (2.1), Transaction Monitoring (2.2), or Terminal Monitoring (2.3) submenus when you set monitoring, breakpoints, or other options
- When using transaction-based commands, enter the USR= option with any CNTL monitoring command

If you override the default user ID setting before you set monitoring, make sure to use the same user ID when you set a breakpoint or other monitoring option. Otherwise, the breakpoints or other monitoring options are not attached to the original monitoring entry. View the Monitoring Status report to check the entries.

Establish a Personal Debugging Session

In a secure CICS region, set the user ID to your own CICS user ID to establish a personal debugging session. When you set monitoring, specify the program or transaction and specify your own CICS user ID. The monitoring request is then invoked only when you execute the program. In addition, specify your own CICS user ID when you set breakpoints and other monitoring options. When you do this, the monitoring request and any breakpoints and options are automatically directed to any terminal you are signed on to under CICS.

Override the To and From Terminal Defaults for Breakpoints

The From and To terminal values on the Monitoring Status display specify where breakpoints take effect and appear. If you are monitoring under a specific CICS user ID, the default From and To terminal values are set to .ANY. If you are monitoring under .ANY user, the default From and To terminal values are set to the terminal where the breakpoints were set.

If you have special testing needs, set breakpoints that include a From and To terminal value that you specify. To do this, set your breakpoints using one of these methods:

- Program, Transaction, or Terminal Monitoring menus (2.1, 2.2, or 2.3) as discussed in the chapter "Using Breakpoints"
- CNTL transaction-based commands as discussed in the chapter "CNTL: Commands and Menus"

Monitoring Submenus

This section describes how to use the Monitoring submenus and screens to perform the following monitoring-related functions. The order in which the functions are described corresponds to their order on the Monitoring Programs, Transactions, and Terminals submenus.

Setting and Removing Monitoring

Begin and end monitoring for programs, transactions, or terminals.

Status Display

Display the monitoring status of a program, transaction, or terminal.

Using the New Copy Option

Reset monitoring for a recompiled program.

Setting and Removing Replacement Options

Dynamically change CICS resources (programs, files, transient data queues, temporary storage).

Setting and Removing Protection Options

Bypass CA InterTest for CICS rules that protect main storage, the CSA, and load modules.

Setting and Removing Special Options

Modify monitoring rules (for example, prevent a program from updating a file, let a program to branch to another program, limit the number of times a program is monitored).

Setting and Removing Composite Support

Provide symbolic support for a program consisting of separately compiled modules.

Viewing and Setting System-Wide Options

Request and revise monitoring options on a system-wide basis.

For more information:

Using Breakpoints (see page 103)
Breakpoint Activities (see page 135)

Monitor Options for Programs, Transactions and Terminals

From the Monitoring Programs, Monitoring Transactions, and Monitoring Terminals submenus, set the following options for a monitored program, terminal, or transaction:

Monitoring Option	Description
Replace	Dynamically replace one or more CICS resources (programs, files, temporary storage, transient data queues) used by a program.
Protect	Override default protection rules.
Special	Set and remove special monitoring options (such as limiting the number of times a program is monitored or preventing a program from updating a file).
Composite	Provide symbolic support for programs consisting of separately compiled modules brought together when the program is link-edited (composite support). Available from the Monitoring Programs submenu only.

Initiate Monitoring

The CA InterTest for CICS CNTL transaction performs all monitoring functions. However, CA InterTest for CICS provides a number of ways to initiate monitoring and perform other CNTL functions as shown following:

- Source Listing Facility
- Monitoring Menu, Option 2 on the Primary Option Menu
- CNTL menus and fill-in screens
- CNTL commands

Use one method exclusively, or mix-and-match methods (even during one test session) depending on your specific testing needs.

Example

Use the Source Listing facility to initiate monitoring and set breakpoints, the Monitoring Menu and screens to set special options, and a single-line command to resume execution of a task.

The easiest way to turn on monitoring and set breakpoints for a single program is to use the Source Listing facility. For a new or infrequent user, this method is strongly recommended. Turn on monitoring by pressing a PF key, and set a breakpoint by entering a single keystroke directly on your program's source listing.

To take advantage of the full range of CA InterTest for CICS monitoring capabilities, use the Monitoring Menu, which is Option 2 on the Primary Option Menu. Additional submenus guide you through selecting monitoring for programs, terminals, or transactions, with or without monitoring options.

Online help is available from every screen. Press **PF1** from any menu or panel for details on how to use it.

If you prefer, enter single-line CNTL commands to specify monitoring functions. More experienced users often find it faster to enter commands directly.

When you **set** breakpoints, replacement, protection, special options, or composite support, you automatically set monitoring for the programs, transactions, or terminals that you specify. So, for example, it is not necessary to set monitoring and set breakpoints. Simply setting breakpoints accomplishes both functions.

However, when you **remove** breakpoints, replacement, protection, special options, or composite support, you do not remove monitoring for the programs, transactions, or terminals that you specify. Monitoring remains in effect until you explicitly instruct CA InterTest for CICS to remove monitoring.

For more information:

Source Listing Facility (see page 47)

The Program Monitoring Submenu

The Program Monitoring submenu (2.1) packs many functions into a single display, but is simple to use. Use this submenu to view, set, or remove monitoring, breakpoints, and monitoring options for programs.

```
        --------CA InterTest r8.5   PROGRAM MONITORING--------------

COMMAND ===>

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

   Program  . . cob*____     Program name (or .ALL, .OPTIONS or generic)
   User ID  . . .ANY____     User (or .ANY) for whom the program is monitored

   Option       Description                                          More:   +
_  Status       Display and/or remove monitoring options (S only)
s  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
_  Mon          Segmented Monitoring On option
_  Nom          Segmented Monitoring Off option

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

How to Use

To select an option, enter or modify the Program and User ID field values, and then tab to the desired Option. Type an **S** to set the option or an **R** to remove the option, and then press Enter to process the function. Press **PF1** for field level help at any time.

Scroll the Options List

The More field on the right side of the display indicates additional options are available. More: + indicates you can scroll forward to view additional options. More: - indicates you can scroll backward to view additional options. Press PF8 to scroll forward and view additional options; PF7 to scroll backward and view previous options.

After you process an option that requires an additional screen, when you return to this submenu the option you processed becomes the first option on the list. Enter the Command **TOP** to display the Options list from the top. Press PF7 to scroll backward, PF8 to scroll forward.

Note: If a program is stopped at a breakpoint, press PF6 to access the Breakpoint Menu, and then select option 1.2.1 to access the Program Monitoring submenu. Jump to the menu by entering =1.2.1 in the command line of the breakpoint display.

Generic Specification of Programs, Transactions, or Terminals

Use the Monitoring submenus or CNTL transaction-based commands to set monitoring or monitoring options generically. Generic specification in CA InterTest for CICS lets you specify asterisks, *, and plus signs, +, to replace characters in a program name, transaction code, or terminal name. These specifications are interpreted according to the rules for the CICS CEMT transaction, as follows:

- An asterisk specifies any number of characters, including no characters. Specify more than one asterisk in a program name, transaction code, or terminal name. However do **not** specify an asterisk by itself.
- A plus sign specifies one character in a particular position. Specify more than one plus sign in a name or code. However do **not** specify a plus sign by itself.
- Combine asterisks and plus signs.

Examples

The following table shows how the specifications are processed:

Generic Entry	Description
ABC*	Specifies all names or codes that begin with ABC, including ABC.
*ABC	Specifies all names or codes that end with ABC, including ABC.
AB*C	Specifies all names or codes that begin with AB and end with C and that could have an unspecified number of characters in the middle.
ABC+	Specifies all four-character names or codes that begin with ABC followed by any one character.
+ABC	Specifies all four-character names or codes that begin with any character followed by ABC.
AB+C	Specifies all four-character names or codes that begin with AB followed by any one character and C.
AB+C*	Specifies all names or codes that begin with AB followed by one character, C, and an unspecified number of characters.

Monitor All Programs

CA InterTest for CICS enables you to establish system-wide monitoring of all applications in the CICS region except for those with the prefix DFH (IBM CICS programs) or IN25 (CA InterTest for CICS programs). Establish system-wide monitoring that applies to .ANY user in the CICS region or a specific CICS user. Typically, this option is password protected.

Menu Entries

To set monitoring for all programs, enter the following on the Program Monitoring menu (2.1):

Program

.ALL

User ID

.ANY to apply system-wide monitoring for all users or **specific user ID** to apply monitoring to all programs executed by a specific user.

Type **S** next to Monitor and press **Enter**.

Hierarchy Rules for Monitored Entries

It is important to know the priorities CA InterTest for CICS observes if more than one entry in the Monitoring Table applies to the same program.

- A specific or generic program entry overrides a transaction entry, a terminal entry, and an .ALL specification
- A transaction entry overrides a terminal entry and an .ALL specification
- A terminal entry (specific or generic) overrides an .ALL specification

Example

Suppose you specify monitoring options for program ABC1 and different options for transaction TRN1. ABC1 executes as part of that transaction. When ABC1 executes, the options specified for ABC1 override the options specified for TRN1. Similarly, it is important to understand the priorities CA InterTest for CICS observes when a program, transaction, or terminal is specifically and generically declared for monitoring. A specific entry overrides a generic entry *except* when you request Status Display. In that case, the generic specification takes precedence.

Suppose you specify monitoring options for program ABC1 and different monitoring options for program ABC*. The options specified for ABC* affect all programs whose names begin with ABC except for program ABC1 (and any other programs beginning with ABC that have their own entries in the Monitoring Table). Monitoring for ABC1 is controlled by the specific entry for that program. However, if you request a Status Display for program ABC*, you get monitoring status reports for all programs that begin with ABC, including program ABC1 that has a separate Monitoring Table entry.

Monitoring

Set and remove monitoring for programs, transactions, or terminals directly from the Monitoring Submenus for Program, Terminals, or Transactions.

Set Monitoring

When you set monitoring for a:

- Program, CA InterTest for CICS monitors the program from whatever terminal it executes, even if it executes as part of a non-terminal owning task. Use the replacement, protection, or special options screen if you want to restrict monitoring to a specific terminal.
- Transaction, CA InterTest for CICS monitors all of the transaction's programs.
- Terminal, CA InterTest for CICS monitors every program that executes at that terminal.

Note: CA InterTest for CICS does not monitor programs that begin with the prefix DFH (IBM CICS programs) or IN25 (CA InterTest for CICS programs).

If you set monitoring for a transaction or terminal, and a program executing as part of that transaction or at that terminal is stopped at an automatic breakpoint, CA InterTest for CICS automatically creates an individual entry in the Monitoring Table for that program. This lets you set breakpoints and monitoring options specifically for that program. Similarly, if you specify .ALL in the Program Names field on the Function Selection Menu or if you specify a generic program name, CA InterTest for CICS creates an individual Monitoring Table entry for any program monitored under that entry that it stops at an automatic breakpoint.

Remove Monitoring

When you remove monitoring, you also remove all breakpoints and monitoring options specified for that program, transaction, or terminal. A quick way of creating a new testing scenario for a program is to remove monitoring and reset it with different breakpoints and options.

Before you remove monitoring for a program, terminal, or transaction being monitored under .ANY user ID, you might want to view the status report of all CA InterTest for CICS monitoring activity to make sure other users are not currently testing the same programs. Option 2.4 gives a Monitoring Status report of all monitoring activity.

From the Status Report

From the status report, remove monitoring quickly for any entry. Just locate the monitoring entry, enter **r** in the field to the left of the entry, and press **Enter**. The following screen shows how to remove monitoring for the program COBDEMO from the Monitoring Status display.

```
----------------CA InterTest r8.5   MONITORING STATUS-----------------
 COMMAND ===>

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

     Option     Description               Attributes
 r  COBDEMO    Program monitor entry      COBOL
 _  |.ANY      User monitoring options    Active
 _     |        Symbolic listing file     PROTSYM
 _     |SLB    Source listing breakpoints X507
              *** End of data ***
```

After the request is processed, an asterisk in front of the Monitoring Status entry indicates the entry was removed. If you press PF2 Refresh, the entry is deleted from the display.

From the Menus

To remove monitoring for programs, transactions, or terminals using the Monitoring submenus, access the appropriate submenu from the Monitoring Menu, and then enter the program name, transaction code, or terminal name, the user ID for whom the program is monitored, and **R** next to the Monitor Option. Press Enter.

Note: Remember that you can also set monitoring by setting breakpoints, replacement, protection, special options, or composite support. However, if you remove breakpoints, options, or composite support, monitoring *remains in effect* for the specified programs until you specifically instruct CA InterTest for CICS to stop monitoring.

Set and Remove Statement Trace Options

Note: These options are unrelated to the backtrace facility.

Set and remove statement tracing and data monitoring for programs, transactions or terminals. However, these options are only valid for COBOL programs and, when specified for a transaction or terminal, will only take affect when a COBOL program is executed. In addition, if either option is specified for more than one program in a transaction, only one statement trace table will be allocated for the transaction.

When you set statement tracing for a COBOL program, if a statement trace table was not created for the transaction, it is created prior to executing the next statement in the program. If a statement trace table exists for the transaction, it will be used by all programs in the transaction for which statement tracing was specified.

When you set statement tracing for a transaction, a statement trace table is created prior to executing the next COBOL statement in the transaction. When you set statement tracing for a terminal, a statement trace table is created for each transaction that executes a COBOL program at that terminal.

When you set data monitoring for a program, transaction, or terminal, statement tracing is automatically set for that entity. Unlike statement tracing, however, data monitoring must be restricted to a specific terminal or user ID.

To set or remove statement trace options, access the appropriate Monitoring submenu:

Option	Used For Monitoring
2.1	Programs
2.2	Transactions
2.3	Terminals

Set Statement Trace Options

On the Monitoring submenu, enter the program name, transaction code, or terminal name for which the option will be set. If the option will be set by user ID, specify this. Select the **STMT Trace** option, using **S,** and press Enter. The Statement Trace Options screen opens.

```
        CA InterTest MONITORING COMMAND BUILDER – STATEMENT TRACE OPTIONS  16

    SET statement trace options for: PROG=COBDEMO

        Enter X next to each option desired:
             Statement tracing (TRC) _
             Data monitoring   (DM ) _

      Term ID (or .ANY or .NO) where trace options will take effect:
      User ID (or .ANY) who will execute the program:                  _______

  Note: If data monitoring is selected either a specific terminal ID or user ID
        is required.




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

Specify the following options on the Statement Trace Options screen. The equivalent option used in CNTL transaction-based commands is specified in parentheses.

Statement tracing (TRC) —Instructs CA InterTest for CICS to save trace information for every executed COBOL statement.

Data monitoring (DM) —Instructs CA InterTest for CICS to capture data values for every executed COBOL statement. If data monitoring is selected, statement tracing will automatically be activated. In addition, a specific terminal ID or user ID will be required.

Term ID (or .ANY or .NO) where trace options will take effect —Specifies that the program be monitored and that statement trace options, if specified, take effect only when the program is executed at any terminal (.ANY), no terminal (.NO), or a specific terminal.

User ID (or .ANY) who will execute the program —Specifies that the program be monitored and that statement trace options, if specified, take effect only when the program is executed by any user (.ANY) or a specific user (a CICS user ID).

Once you have specified the required information, press Enter. You are returned to the monitoring submenu where you selected the Statement Trace option.

Remove Statement Trace Options

This section describes how to remove statement trace options from a status report or from the menu.

From a Status Report

To remove statement tracing, access the Monitoring Status display (2.4), scroll to the monitoring entry for which you specified the option, and locate the statement trace option. Type **r** next to the option and press Enter. If data monitoring is on, it is also removed.

```
-------------------- CA InterTest r8.5   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      COBOL
_ - |-.ANY     User monitoring options    Active
      |        Symbolic listing file      PROTSYM
R -   |-TRC    Statement trace active      X507
      |        Option ID                  DD180934
      |        Data Monitoring            ON
_     |-SLB    Source listing breakpoints X507
               *** End of data ***
```

From the Menus

Alternatively, use the same menus and screens you used to set the option, but instead of entering **s** next to Trace, enter **r** next to Trace.

To remove statement tracing, enter **R** next to the Statement Trace option. If data monitoring is on, it is also removed. To remove data monitoring without removing statement tracing enter **x** next to Data monitoring. Enter the terminal ID and user ID exactly as you originally defined them.

New Copy Option

After you recompile a program, use the New Copy option on the Program Monitoring menu (2.1) to reset symbolically specified breakpoints and other monitoring options for a recompiled program. New Copy also resets the entry in the PPT to the program's new library address.

Note: CNTL=NEW is the equivalent CNTL command for the New Copy monitoring menu option.

Replacement Options

This section explains how to set and remove replacement options. Set replacement options to dynamically change the names of CICS resources (programs, files, transient data queues, and temporary storage) specified in CICS CALLS by a monitored program. Remove the replacement options when you want the program to use the resources defined in the program.

When you specify replacement options for a transaction, the resources are replaced in all of its programs. When you specify replacement options for a terminal, the resources are replaced in all programs executing from that terminal.

To set or remove replacement options, access the appropriate Monitoring subenu to which the monitoring option applies:

Option	Used For Monitoring
2.1	Programs
2.2	Transactions
2.3	Terminals

Set Replacement Options

On the Monitoring submenu, enter the program names, transaction codes, or terminal names and the user ID option for whom the entry will be monitored. Then select the **Replace** option using **S,** and press Enter. The Replacement Options screen appears.

```
          CA InterTest MONITORING COMMAND BUILDER  REPLACEMENT OPTIONS     20

    SET one or more options to replace one CICS resource name with another in:
                              PROG=COBDEMO


     Replace program name:          ________   with   ________
     Replace file name:             pfile___   with   tfile___
     Replace TD queue name:         ____       with   ____
     TS selection mask:             ____________________________________
     TS replacement mask:           ____________________________________

     Limit monitoring to your terminal  '*'  or TERMID:       ____
     User ID (or .ANY) who will execute the program:           ________



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

Specify the following options on the previous screen. The equivalent option used in CNTL transaction-based commands is specified in parentheses.

Replace program name (RPC)—Lets a program name be replaced. Specify the original program name and the new name.

Replace file name (RFC)—Lets a file name be replaced. Specify the original file name and the new name. The sample screen shown previously specifies that program COBDEMO use file TFILE instead of PFILE.

Replace TD queue name (RTD)—Lets a transient data queue name be replaced. Specify the original TD queue name and the new name.

TS selection mask; TS replacement mask (RTS)—Lets a temporary storage ID be replaced. Specify the original temporary storage ID in the TS selection mask field and the new temporary storage ID in the TS replacement mask field. Specify both masks as eight bytes in either character (Cdata) or hexadecimal (Xdata) format.

Example

STORAGE2 replaces STORAGE1 in all temporary storage requests issued by the program.

```
TS selection mask:   C'STORAGE1'
TS replacement mask: C'STORAGE2'
```

Limit monitoring (TON)—Specifies that the program be monitored and that replacement options, if specified, take effect only when the program executes at the current terminal (*), a specific terminal (termid), or without a terminal (.NO).

User ID (or .ANY) who will execute the program—Specifies that the program be monitored and that replacement options, if specified, take effect only when the program is executed by any user (.ANY) or a specific user (a CICS user ID).

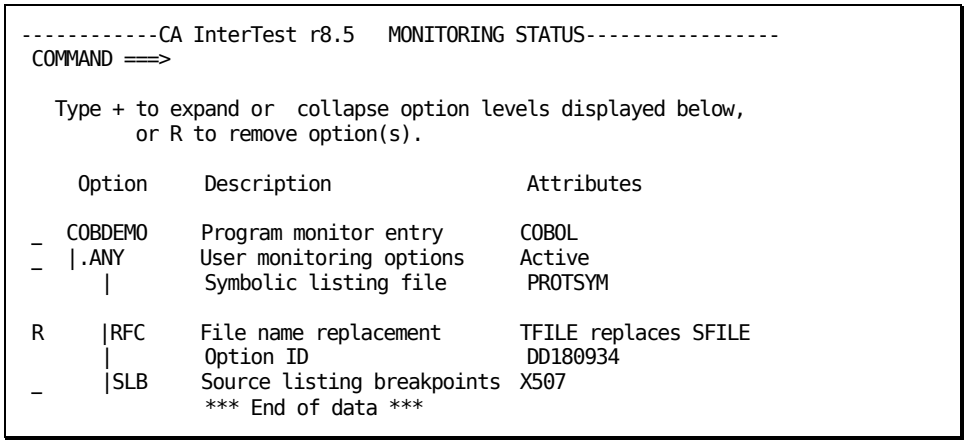
When you have specified the required information, press Enter. You are returned to the monitoring submenu where you selected the Replace option.

Remove Replacement Options

This section describes how to remove replacement options from a status report or from the menu.

From a Status Report

To remove a replacement option, access the Monitoring Status display (2.4), scroll to the monitoring entry for which you specified the option, and locate the specific replacement option. Type **r** next to the option and press Enter.



```
------------CA InterTest r8.5   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      COBOL
 _  |.ANY       User monitoring options    Active
       |        Symbolic listing file      PROTSYM

 R     |RFC     File name replacement      TFILE replaces SFILE
       |        Option ID                  DD180934
 _     |SLB     Source listing breakpoints X507
                *** End of data ***
```

From the Menus

Alternatively, use the same menus and screens you used to set the option, but instead of entering **s** next to Replace, enter **r** next to Replace. Specify the information exactly as you entered it.

To remove all replacement options of a specific type, use the menus and enter **.ALL** in the appropriate field on the Remove Replacement Options screen as shown in the following screen. This example removes all file replacement options previously set for the transaction DEMC.

```
  CA InterTest MONITORING COMMAND BUILDER  REPLACEMENT OPTIONS     20

REMOVE one or more options to replace one CICS resource name with another in:
                              TRAN=DEMC


    Replace program name:         ________   with   ________
    Replace file name:            .all____   with   ________
    Replace TD queue name:        ____       with   ____
    TS selection mask:            ____________________________________
    TS replacement mask:          ____________________________________

    Limit monitoring to your terminal  '*'  or TERMID:       ____
    User ID (or .ANY) who will execute the program:           ________



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

Protection Options

Set options to override the CA InterTest for CICS default protection rules for modifying main storage, the CSA, and load modules. Remove the protection options to re-instate the default protection rules.

Note: Because the protection options can potentially damage your CICS system if misused, exercise caution in specifying them. These options are password protected unless your site removed this restriction.

To set or remove protection options, access the appropriate Monitoring submenu to which the monitoring option applies:

Option	Used For Monitoring
2.1	Programs
2.2	Transactions
2.3	Terminals

Set Protection Options

On the Monitoring submenu, enter the program names, transaction codes, or terminal names and the user ID option for whom the entry will be monitored. Then select the **Replace** option using **s,** and press Enter. The following Protection Options screen appears.

```
CA InterTest MONITORING COMMAND BUILDER  PROTECTION OPTIONS       21

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


     Bypass storage protection:      From    ________   To       ________
     Unprotect CSA:                  Offset  ____       Length   ___
     Unprotect CWA:                  Offset  ____       Length   ___
     Unprotect main storage area:    Address ________   Length   _____
     Unprotect load module:          Name    ________
     Protect main storage area:      Address ________   Length   _____



     Limit monitoring to your TERMINAL  '*' or TERMID:          ____
     user ID (or .ANY) who will execute the program             _____
```

Specify the following options on the previous screen. In the description of each option, the command syntax equivalent for each monitoring option is specified in parentheses.

Bypass storage protection (BYP)

Permits a specified section of program code to do the following:

- Modify any area of main storage
- Issue SVC instructions
- Issue BALR, BASSM, or BASR 14,15 or 14,14 instructions

Specify the beginning and ending program addresses or program offsets. Specify addresses as six- to eight-hexadecimal digits, and offsets as one- to five-hexadecimal digits.

If you specify .ANY in the From field and an address or offset in the To field, CA InterTest for CICS suspends monitoring when any BALR, BASSM, or BASR 14,15 or 14,14 instruction passes control to the specified location. Monitoring resumes when the routine executed by the instruction returns control to the next byte after the instruction.

At a breakpoint, specify * in the From field and leave the To field blank to instruct CA InterTest for CICS to bypass the current instruction.

Unprotect CSA (CSA)

Permits a program to modify areas in the CSA. Specify the offset and the length in hexadecimal.

Unprotect CWA (CWA)

Permits a program to modify areas in the CWA. Specify the offset and the length in hexadecimal.

Unprotect main storage area (LET)

Permits a program to modify a designated area of storage. Specify the beginning address as six to eight hexadecimal digits. Specify the length of the area in hexadecimal.

Unprotect load module (LET)

Permits a program to modify a load module.

Protect main storage area (PRO)

Prevents a program from modifying a designated area of storage not protected by CA InterTest for CICS. Specify the beginning address as six- to eight-hexadecimal digits. Specify the length of the area in hexadecimal.

Limit monitoring (TON)

Specifies that the program be monitored and that the protection options, if specified, take effect only when the program executes at the current terminal (*), a specific terminal (termid), or without a terminal (.NO).

User ID (or .ANY) who will execute the program (USR)

Specifies that the program be monitored and that protection options, if specified, take effect only when the program is executed by any user (.ANY), or a specific user (a CICS user ID).

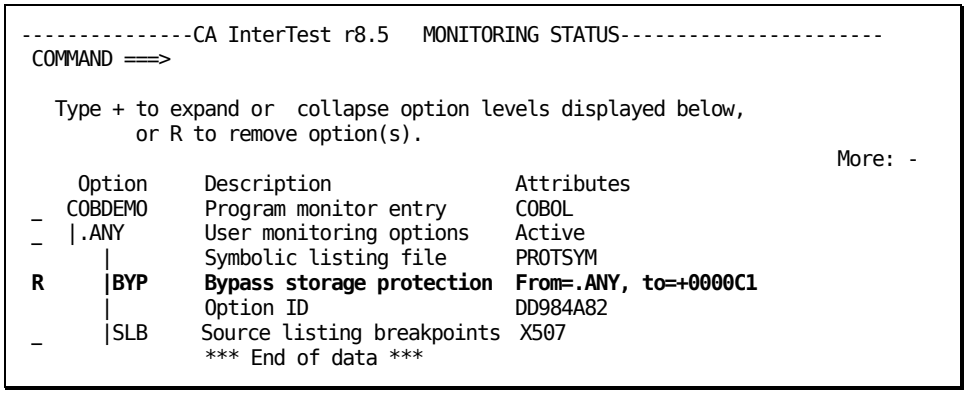
Once you have specified the required information, press Enter. You return to the monitoring submenu where you selected the Protect option.

Remove Protection Options

This section describes how to remove protection options from a status report or from the menu.

From a Status Report

To remove a protection option, access the Monitoring Status display (2.4), scroll to the monitoring entry for which you specified the option, and locate the specific protection option. Type **r** next to the option and press **Enter**.



```
---------------CA InterTest r8.5   MONITORING STATUS----------------------
 COMMAND ===>

   Type + to expand or  collapse option levels displayed below,
          or R to remove option(s).
                                                                    More: -
     Option     Description               Attributes
_   COBDEMO    Program monitor entry     COBOL
_   |.ANY      User monitoring options   Active
       |       Symbolic listing file     PROTSYM
R      |BYP    Bypass storage protection From=.ANY, to=+0000C1
       |       Option ID                 DD984A82
_      |SLB    Source listing breakpoints X507
              *** End of data ***
```

From the Menus

Alternatively, to remove protection options, access the Monitoring submenu, specify monitoring entry, (program, transaction, or terminal and user ID), type **R** next to the Replace option and press Enter. Then enter the information exactly as you originally defined it.

To remove all options of a specific type, enter **.ALL** in the appropriate field of the Protection Options menu.

Example

To remove all Bypass Storage Protection options, enter **.ALL** in the first Bypass Storage Protection field.

Special Options

Set special options to alter the standard CA InterTest for CICS monitoring procedures. Remove the special options to reinstate the default monitoring rules.

To set or remove protection options, access the appropriate Monitoring submenu to which the monitoring option applies:

Option	Used For Monitoring
2.1	Programs
2.2	Transactions
2.3	Terminals

Set Special Options

On the Monitoring submenu, enter the program names, transaction codes, or terminal names and the user ID for whom the entry will be monitored. Then select the **Special** option using **S,** and press Enter. The following Special Options screen appears.

```
          CA InterTest MONITORING COMMAND BUILDER - SPECIAL OPTIONS        22

     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) _______
          Limit total number of CICS requests  (MXR) _______
          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:             ________


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

You can specify many options on this screen. The equivalent option used in CNTL transaction-based commands is specified in parentheses.

Source Listing Breakpoint (SLB)—Specifies that the Source Listing Breakpoint screen be displayed rather than the detailed breakpoint display screen.

No file updating (NUP)— Specifies that a monitored program not update any files. This option only affects VSAM and BDAM files; databases are unaffected.

Reentrancy check (RNT)—Prevents a program from modifying its own code.

Follow monitoring (FOL)—Instructs CA InterTest for CICS to continue monitoring a program even if it branches directly to another program (wild branch). All monitoring options remain in effect. Specify the following:

- ON when control passes to a program that does not have a PPT entry
- The name of the program when control passes to a program that has a PPT entry
- NOPPT instead of ON to reduce overhead when a program often branches to a program that does not have a PPT entry

Note: The FOL option should not be specified if control is passed to a program by a LINK or XCTL instruction. Such a program is monitored only if monitoring was specified for it.

Number of times to be monitored (MUS)—Limits the number of times a program should be monitored.

Limit total size of CICS storage (MXS)—Limits the amount of storage, including program storage, a program acquires.

Limit total number of CICS requests (MXR)—Limits the number of requests a transaction issues. This option is useful when a program is in a loop that includes a CICS request.

Setting for Structure Display Format (SDF)—Use this option to override, by program, the global default setting. Specify the following:

HEX to display data in hexadecimal / character format.

DATA to display data in Structure Display Format

Set local automatic breakpoint (ABP)— Activates the automatic breakpoint facility for a particular program, transaction, or terminal.

Specify the terminal to receive automatic breakpoint displays. An asterisk, *, routes the display to the current terminal. A termid routes the display to a particular terminal. .ANY routes the display to all terminals. OFF instructs CA InterTest for CICS to abend the task instead of halting it at an automatic breakpoint.

Note: This option is useful when you do not want the breakpoint display routed to the terminal where the program is running or when the display **cannot** be routed to that terminal (for example, a non-terminal task or a task running on a non-3270 terminal). In those cases, specify the terminal to which breakpoint displays should be routed. The local automatic breakpoint option is also useful in a production environment when the **global** Automatic Breakpoint facility is disabled and you want to monitor a specific program, transaction, or terminal.

Limit monitoring (TON)—Specifies that the program be monitored and that the special options, if specified, take effect only when the program executes at the current terminal (*), at a specific terminal (termid), or without a terminal (.NO).

User ID (or .ANY) who will execute the program (USR)—Specifies that the program be monitored and that special options, if specified, take effect only when the program is executed by any user (.ANY) or a specific user (a CICS user ID).

Once you have specified the required information, press **Enter**. You are returned to the monitoring submenu where you selected the Protect option.

For more information:

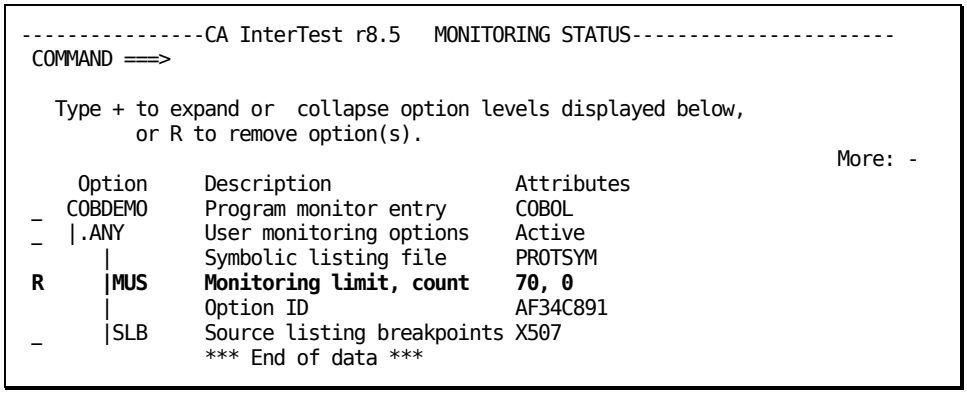
Source Listing Facility (see page 47)

Remove Special Options

This section describes how to remove special options from a status report or from the menu.

From a Status Report

To remove a special option, access the Monitoring Status display (2.4), scroll to the monitoring entry for which you specified the option, and locate the specific special option. Type **R** next to the option and press Enter.



```
----------------CA InterTest r8.5   MONITORING STATUS----------------------
 COMMAND ===>

   Type + to expand or  collapse option levels displayed below,
          or R to remove option(s).
                                                                    More: -
     Option     Description               Attributes
_   COBDEMO    Program monitor entry     COBOL
_   |.ANY      User monitoring options   Active
       |       Symbolic listing file     PROTSYM
R      |MUS    Monitoring limit, count   70, 0
       |       Option ID                 AF34C891
_      |SLB    Source listing breakpoints X507
              *** End of data ***
```

From the Menus

To remove special options, access the same monitoring menu and specify **R** next to Special, complete the Special Options menu by enter an **x** next to the options you want to remove, or enter the information exactly as you originally defined it.

Composite Support

The CA InterTest for CICS composite support feature lets you take advantage of all CA InterTest for CICS capabilities when you test and debug composite modules. A *composite module* is a load module, defined in the PPT that consists of separately compiled or assembled parts brought together when the module is link-edited. The part of the composite module that receives control from CICS is referred to as the *main program*. The remaining parts are referred to as *subprograms*. The main program and subprograms can be written in the same or different languages.

Composite support lets you test and debug a subprogram as if it were a separate program. This means you can set breakpoints and other monitoring options individually for any subprogram. If the same subprogram is defined to more than one composite load module, see the section Composite Support for Duplicate Subprograms (see page 98) in the chapter "Source Listing Facility" for more details.

CA InterTest for CICS provides **full symbolic support** for the main program and subprograms if you provide that information when the programs are compiled or assembled.

CA InterTest for CICS needs **link-edit** information for the main program and subprograms you want to test. There are two ways to provide link-edit information as shown following:

- Execute a batch job
- Enter specifications online on the Composite Support screen

Note: The batch method is recommended because it is more efficient.

For more information:

Source Listing Facility (see page 47)

Set Composite Support Using the Batch Method

To set composite support using the batch method, perform these steps:

1. Execute a batch job.
2. Access the Composite Support screen.
3. Press **PF5** to process the specifications.

The CA InterTest for CICS batch job step, executed after the composite module is link-edited, reads the link-edit map to determine the link-name, offset, and length of the main program and each subprogram. This step also specifies the names under which CA InterTest for CICS monitors each program.

After the batch job completes, access the Program Monitoring menu (2.1). Enter the composite module name, the user ID who executes the program (or .ANY), type **S** next to the Composite Option, and press Enter. CA InterTest for CICS displays the following Composite Support screen.

```
           CA InterTest MONITORING COMMAND BUILDER - COMPOSITE SUPPORT       23
                    Define composite support for BIGMOD

  Link-name  Monitor-name  Offset  Length  Language            Comments

_ MAINMOD      BIGMOD        160    78A8  COBOL
_ SUBMOD-A     ASMMOD-A     7CA0    1200  ASSEMBLER
_ SUBMOD-B     ASMMOD-B     8EA0    1001  ASSEMBLER
_ SUBMOD-C     ASMMOD-C     9EA1     93A  ASSEMBLER
_ SUBMOD-D     ________     A8DB     A11  __________ symbolics not available




Add, remove, or change monitor names as needed, and press PF5
CAIN2602 BOTTOM of data
PF1 Help       2           3 End        4 Return     5 Process    6
PF7 Backward   8 Forward   9           10           11           12
```

The Composite Support screen lists link-edit information for the main program and subprograms. Subprograms with prefixes DFH, ILB, and IBM are automatically excluded from the list unless you specifically name them in the batch job step.

Monitor-Names

Make the following changes on this screen:

- Type monitor-names for subprograms that you want to test that do not have monitor-names already specified
- Blank out monitor-names of subprograms you **do not** want to test to reduce overhead
- Change the monitor-name and language specified for a subprogram

Once you have made all necessary changes, press **PF5** to set composite support.

The fields on the Composite Support screen provide the following information:

Link-name—Specifies the name of a COBOL program, Assembler CSECT, or PL/I Control Section as listed in the link-edit map.

Monitor-name—Specifies the name under which CA InterTest for CICS monitors the program. The following rules apply to monitor-names:

- Each monitor-name must be unique.
- The monitor-name of the main program will always be the PPT name of the composite module. This name **cannot** be changed.
- The monitor-name of a subprogram **cannot** be a PPT name.
- A monitor-name is identical to a link-name.
- If you want symbolic information for a program or subprogram, its monitor-name must be identical to the name used in the post-processor batch job step that provides the symbolic information to CA InterTest for CICS.

Example

In the previous figure BIGMOD is the monitor-name of the main program; ASMMOD-A, ASMMOD-B, and ASMMOD-C are the monitor-names of the subprograms you want to test.

Offset—Specifies the hexadecimal offset of the program as listed in the link-edit map. If the offset displays on the screen, you cannot change it.

Length—Specifies the hexadecimal length of the program as listed in the link-edit map. If the length displays on the screen, you cannot change it.

Language—Specifies the language in which the program is written. If CA InterTest for CICS has symbolic information for the program, it uses the language specified in the post-processor batch job. If CA InterTest for CICS does not have symbolic information, it assumes the subprogram is written in the same language as the main program.

If necessary, change this field by entering:

COB or COBOL for COBOL
COB2 or COBOL II for COBOL II
COB/370 or COBOL/370 for COBOL/370
IBMCOB for COBOL for MVS and z/OS
ASM or HLASM for Assembler
PLI or PL/I for PL/I

Comments—Specifies optional comments, which describe symbolic information available for the program:

symbolics not available—CA InterTest for CICS does not have symbolic information for the program.

symb data after linkedt—The program might have been recompiled or reassembled after the link-edit. Composite support information might be outdated.

lang chg to match symb—The language has been changed to the language specified in the symbolic post-processor batch job.

Use PF8 and PF7 to scroll forward and backward through the list of programs. Enter a slash, /, to the left of a link-name to position that program on the top of the next screen.

Change the Composite Support Screen

Add, delete, or change monitor-names of subprograms to meet your testing needs.

Example

Suppose CA InterTest for CICS displays the Composite Support screen for composite module BIGMOD. You want to test SUBMOD-D, but you do not want to test SUBMOD-A, SUBMOD-B, and SUBMOD-C. To do this, change the monitor-name specifications, as shown, before pressing PF5.

```
           CA InterTest MONITORING COMMAND BUILDER - COMPOSITE SUPPORT       23
                    Define composite support for BIGMOD

   Link-name  Monitor-name  Offset  Length  Language            Comments

_ MAINMOD       BIGMOD        160     78A8  COBOL
_ SUBMOD-A                   7CA0     1200  ASSEMBLER
_ SUBMOD-B                   8EA0     1001  ASSEMBLER
_ SUBMOD-C                   9EA1      93A  ASSEMBLER
_ SUBMOD-D      ASMMOD-D     A8DB      A11  __________ symbolics not available









Add, remove, or change monitor names as needed, and press PF5
CAIN2602 BOTTOM of data
PF1 Help       2            3 End        4 Return     5 Process    6
PF7 Backward   8 Forward    9           10           11           12
```

Example

- The monitor-names for SUBMOD-A, SUBMOD-B, and SUBMOD-C were removed because those subprograms are not going to be tested. Removing the monitor-names reduces overhead.
- A monitor-name was specified for subprogram SUBMOD-D so that the subprogram is tested separately.

Note: Changes made to the Composite Support screen affect that session only. So, if you re-accessed the Composite Support screen for BIGMOD after making the previous changes, the original screen is displayed. You could then submit the original specifications or change those specifications again.

Set Composite Support Using the Online Method

The online method of setting composite support consists of the following steps:

1. Access the Composite Support screen.
2. Enter link-edit information.
3. Press **PF5** to process the specifications.

To access the Composite Support screen, perform these steps:

1. Access the Program Monitoring Menu (2.1) and enter the composite module name in the Program name field.
2. Type an S next to the Composite option, and press **Enter**.

```
          CA InterTest MONITORING COMMAND BUILDER - COMPOSITE SUPPORT       23
                    Define composite support for COBDEMO

   Link-name  Monitor-name  Offset  Length  Language            Comments

_ COBDEMO      COBDEMO     000020   0304E  IBMCOB 3.2









Add, remove, or change monitor names as needed, and press PF5
CAIN2602 BOTTOM of data
PF1 Help       2           3 End        4 Return     5 Process    6
PF7 Backward   8 Forward   9           10           11           12
```

Enter the following information for COBDEMO and each subprogram you want to test **separately**:

- **Link-names** of the main program and each subprogram you want to test (required).
- **Monitor-names** for each subprogram you want to test (required).
- Hexadecimal **offset** and **length** of the main program and each subprogram you want to test (required).
- **Language** of the main program and each subprogram you want to test (optional). If CA InterTest for CICS has symbolic information for the programs, omit this field because CA InterTest for CICS uses the language specified in the symbolic post-processor batch job. However, if you omit the subprogram language and CA InterTest for CICS does **not** have symbolic information for that program, it defaults to the language of the main program.

To complete this screen, perform these steps:

1. Enter the link-name, offset, and length specifications as listed on the link-edit map. Observe the rules for monitor-names as described earlier in this section.
2. After entering the necessary information, press **PF5** to process the specifications.

Note: Remember, it is not necessary to enter information for subprograms you do not plan to test separately.

Set Monitoring for a Program with Subprograms

If you set monitoring for a program for which CA InterTest for CICS has link-edit information, a message prompts you to press **PF3** to access the Composite Support screen. This screen contains the link-edit information. Press **PF5** to process the specifications.

Assembler Programs with Multiple CSECTs

For Assembler composite modules, treat multiple CSECTs that were assembled together as one subprogram. Enter a plus sign, +, in the field to the left of the link-names, as shown next.

```
           CA InterTest MONITORING COMMAND BUILDER - COMPOSITE SUPPORT       23
                    Define composite support for ASMLINK

   Link-name  Monitor-name  Offset  Length  Language            Comments

 _ ASMLINK      ASMLINK        48    2000  ASSEMBLER
 _ ASM-A        ASM-A        2048     68B  ASSEMBLER
 + ASM-B        ________     2F98     1BC  ASSEMBLER
 + ASM-C        ________     3B00     290  ASSEMBLER
↗


 Add, remove, or change monitor names as needed, and press PF5
 CAIN2602 BOTTOM of data
 PF1 Help       2            3 End        4 Return     5 Process    6
 PF7 Backward   8 Forward    9           10           11           12
```

In the previous figure, subprograms ASM-A, ASM-B, and ASM-C are treated as one program. Set breakpoints and other options for all CSECTs under the name ASM-A, the monitor-name specified for subprogram ASM-A.

Note: The Composite Support screen might already have plus signs joining Assembler CSECTs. This means the CSECTs were treated as one program in the post-processor batch job step that provided symbolic information to CA InterTest for CICS.

Remove Composite Support

This section describes how to remove composite support from a status report or from the menu.

From a Status Report

To remove composite support, you must remove monitoring for the main program, as shown in the following steps:

1. Access the Monitoring Status Report (2.4).
2. Scroll to the monitoring entry for the main program, type R to the left of the entry, and press **Enter**.

Note: Remember, removing monitoring from the main program also removes monitoring from all subprograms. In other words, all breakpoints and options set for the **main program** and all **subprograms** are removed.

From the Menus

To remove composite support from the menus, perform these steps:

1. Access the Program Monitoring menu (2.1).
2. Specify the main program name, tab to the Monitor option, and type **B**. Press **Enter** to remove monitoring for that program.

Disconnect, Reconnect, and Purge Active Tasks

This section describes how to disconnect or reconnect a terminal to CA InterTest for CICS.

Disconnect a Task

To disconnect a terminal from CA InterTest for CICS while at a breakpoint, use Breakpoint Primary Menu Option 7 Disconnect. This option lets non-CA InterTest for CICS or non-CICS functions be performed at the terminal.

Reconnect a Task

To reconnect a terminal to the CA InterTest for CICS breakpoint, use the ITST Monitoring Menu to reconnect a task. Monitoring Menu Option 5 Active-Tasks, lets you reconnect, disconnect, and purge active tasks. From CICS, enter **ITST 2.5** and the following Active Tasks menu appears.

```
---------------CA InterTest r8.5   ACTIVE TASKS-----------------------
 COMMAND ===>

   Type R to reconnect or P to purge tasks below.

   Task         Description               Attributes
 _ 00093        Transaction, program      DEMC, COBDEMO
                Terminal                  X505
                Waiting at breakpoint     SBP since 06:49 p.m.
                Breakpoint location       Offset=0016FA, address=000D9B42
                Breakpoint reason         Singlestepping breakpoint.
                *** End of data ***



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

The Active-Tasks menu lets you type an **R** next to a task you want to reconnect to your terminal, or type a **P** next to a task you want to purge.

Note: MRO environments should consult the *Installation Guide* for reconnection considerations for tested transactions in implicit routing sessions.

View and Set System-Wide Options

Set certain options on a system-wide basis from the Monitoring System-Wide Options submenu (2.6). The following is the System-Wide Options screen.

```
----------------CA InterTest r8.5   SYSTEMWIDE OPTIONS-----------
 COMMAND ===>

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

   Option       Description               Attributes
 _ ABP          Automatic breakpointing   Active
 _ CKPT         Option checkpointing      Inactive
 _ DUMP         Dumping all abends        Inactive
 _ GLOG         Global logging            Inactive
 _ ITTRACE      CA InterTest tracing      Active
 _ PURGE        Breakpointed task purging Inactive
                *** End of data ***
```

Note: These options are password protected unless your site has removed this restriction.

There are six options to select from on this screen. The equivalent option used in single-line commands is specified in parentheses. Type **S** next to the option to select it, and then complete the following screen. In many cases, you enter ON to set an option and OFF to remove it.

ABP Option: Automatic breakpointing sets the Automatic Breakpoint Facility for the entire system. Default. Breakpoint displays are routed to the terminal where the program is running. Specify a terminal ID to indicate where breakpoint displays should be routed for non-terminal attached tasks and for tasks executing from non-3270 terminals.

```
------------ CA InterTest r8.5   ABP OPTION--------------------------
 COMMAND ===>

   Modify below to set or remove the ABP option, then press ENTER.


   ABP status . . ON_         Automatic breakpointing ( ON or OFF ).
   Terminal ID  . ____        Terminal ID to receive breakpoint.
```

If you specify OFF for the ABP option, you might want to use the Special Options from the Program Monitoring submenu (2.1) to set the local Automatic Breakpoint option for one or more programs. The local Automatic Breakpoint option overrides the system Automatic Breakpoint option.

CKPT Option: Option Checkpointing instructs the Checkpoint Recording Facility to periodically record the status of CA InterTest for CICS monitoring so that you can use the restart feature. The first checkpoint occurs when the option is specified. Subsequent checkpoints occur at the end of the interval, specified in hours and minutes in *hhmm* format. The minimal time interval is ten minutes.

```
------------- CA InterTest r8.5   CKPT OPTION --------------------
 COMMAND ===>

   Modify below to set or remove the CKPT option, then press ENTER.


   CKPT status  . OFF         Checkpoint CA InterTest tables ( ON or OFF ).
   Interval . . . ____        Checkpointing interval ( HHMM; minimum 0010 ).
```

DUMP Option: Dumping all abends requests that a dump be produced for all abends.

```
--------------- CA InterTest r8.5   DUMP OPTION ----------------------
 COMMAND ===>

   Modify below to set or remove the DUMP option, then press ENTER.


   Dump status  . OFF         Dump all abends option ( ON or OFF ).
```

GLOG: Global logging requests global logging of CA InterTest for CICS activity, most CNTL commands, and all changes to main storage.

```
-------------- CA InterTest r8.5   GLOG OPTION --------------------
 COMMAND ===>

   Modify below to set or remove the GLOG option, then press ENTER.


   GLOG status  . OFF         Global command logging ( ON or OFF ).
   TD queue . . : CSSL        Log transient data destination queue.
```

ITTRACE Option: CA InterTest for CICS tracing instructs CA InterTest for CICS to write trace records to the CICS trace during monitoring.

Purge breakpointed tasks (PURGE) instructs CA InterTest for CICS to periodically purge tasks halted at a breakpoint for longer than the specified interval. The first purge occurs when the option is specified. Subsequent purges occur at the end of the interval, specified in hours and minutes in this format *hhmm*. The minimal time interval is 20 minutes.

Chapter 8: Special Monitoring Situations

This chapter guides you on how to use CA InterTest for CICS in the following special monitoring situations:

- Production environment monitoring
- Monitor tasks that execute without terminals
- Monitor tasks that execute at non-3270 terminals
- Use segmented monitoring to handle special situations
- Use CA InterTest for CICS with IBM's EDF (EXEC Debugging Facility)

Note: Segmented monitoring is available with CA InterTest for CICS releases 5.*x* only and applies to users of HOGAN Systems software, the MSA Corporation's DCI interface, and LE/370 Condition Handler monitoring.

Special Monitoring Situations

The following sections of this chapter discuss the following special monitoring situations:

- Monitoring programs with LE/370 condition handlers
- Monitoring dynamically called COBOL II and COBOL/370 programs
- Production environment monitoring
- Monitoring tasks that execute without terminals
- Monitoring tasks that execute at non-3270 terminals
- Use segmented monitoring to handle special situations
- Use programmed breakpoints
- CICS/ESA 3.3 FEPI applications monitoring

Segmented monitoring applies to users of HOGAN Systems software and the MSA Corporation's DCI Interface. Programmed breakpoints are used primarily by PL/I users who want to display data areas that CA InterTest for CICS normally cannot display or who do not have symbolic information for their programs.

Monitor Programs with LE/370 Condition Handlers

LE/370 condition handlers do not receive control in the same manner as a normal CICS COBOL program. Therefore, they must be monitored using the CA InterTest Segmented Monitoring and ABI=OFF options.

To monitor COBOL/370 programs containing calls to enable or disable LE/370 condition handlers, perform these steps:

1. Use normal monitoring.
2. Specify ABI=OFF for the main program, and if there is a composite module, for all subprograms.

   Note: Do not specify ABI=OFF for the LE/370 condition handler module if you want CA InterTest for CICS to intercept abends that occur within the LE/370 condition handler. ABI=OFF is needed for all programs except the condition handler to *prevent* CA InterTest for CICS from intercepting the abend *before* the LE/370 condition handler is invoked.

Because LE/370 condition handlers do not receive control the same way a normal CICS program does, you must start and stop monitoring for the condition handler routine or program by using the Segmented Monitoring breakpoint options (+ and -). For example, use the Source Listing facility to display the condition handler routine or program, and then set a start monitoring (+) breakpoint at the first statement number, which causes CA InterTest for CICS to begin monitoring when the condition handler is invoked by LE/370. Set normal breakpoints, such as unconditional, conditional and so on, in the condition handler program. For details on segmented monitoring, Segmented Monitoring to Handle Special Situations (see page 274).

You must set a stop monitoring breakpoint (-) *before* or *at* the GOBACK statement of the condition handler or unpredictable results might occur.

Monitor Dynamically Called COBOL II and COBOL/370 Programs

The monitoring and debugging of dynamically called COBOL II and COBOL/370 programs is handled as follows:

- In the same way as other CICS COBOL II and COBOL/370 programs without necessarily monitoring higher level programs
- Using the FOL=PPT option

Request monitoring for dynamically called COBOL programs through the Source Listing and CNTL facilities the same way you request monitoring for first level CICS COBOL programs.

For more information:

Using the FOL= Option (see page 571)

Production Environment Monitoring

At times, use CA InterTest for CICS to monitor programs running in a production environment.

Example

You might want to monitor recently installed or modified applications until you are sure they are free of errors, or even monitor the entire production system if you have errors that cannot otherwise be isolated.

When you use CA InterTest for CICS in a production environment, you should consider the following items:

- Where to route automatic breakpoint displays
- How many times to monitor a program
- Whether to restrict monitoring to a single terminal or user ID
- What to do if you must monitor the entire system

Control Automatic Breakpoint Displays

When you monitor a program in a production environment, you usually do not want the breakpoint display to appear at the operator's terminal because the information is meaningless. Instead, CA InterTest for CICS either abends the task or routes the Breakpoint Display.

Abend the Task

If you want CA InterTest to abend the task before it damages CICS and writes a transaction dump (INTE), deactivate the global automatic breakpoint facility. This facility instructs CA InterTest for CICS to generate a breakpoint each time it detects an error in a program that it is monitoring.

To deactivate this facility, select System-Wide Options from the Monitoring Menu (Primary Menu Option 2.6), and enter **OFF** in the Automatic Breakpoint field, or enter the command:

```
CNTL=ABP,OFF
```

When the global automatic breakpoint facility is deactivated, you might want to specify the local automatic breakpoint option for certain programs, transactions, or terminals.

Route the Breakpoint Display

To route breakpoint displays to a programmer's terminal, specify the local automatic breakpoint option and direct the breakpoint displays to a programmer's terminal. The local automatic breakpoint option instructs CA InterTest for CICS to generate a breakpoint each time it detects an error in the specified program, transaction, or terminal, and it overrides the global automatic breakpoint facility.

To specify this option, select Special Options from the Monitoring submenus (2.1, 2.2, or 2.3) and specify the programs, transactions, or terminals. Then specify the terminal to which breakpoint displays should be routed. You could also enter the command:

```
CNTL=ON,promid,ABP=termid
```

Control the Number of Times a Program Is Monitored

When a new or modified program goes into production, limit the number of times CA InterTest for CICS monitors it. The MUS option lets you specify the number of times CA InterTest for CICS should monitor a program. To specify the MUS option, select Special Options from the Monitoring submenus (2.1, 2.2, or 2.3) and specify the programs, transactions, or terminals. On the Special Options menu, specify the number of times monitoring should occur in the MUS field. You can also enter this command:

```
CNTL=ON,promid,MUS=nnn
```

Monitor a Program for a Single User ID

When a program fails in the production system, use CA InterTest to monitor it so you can find and correct the error. In a secure CICS region, restricting monitoring to a single user ID means you test the program thoroughly *without* interfering with the work of other users.

All monitoring methods let you specify a single CICS user ID when setting monitoring. This limits monitoring to only that user, but permits the user to go to any terminal. Monitoring by user ID can be the default in your system. If it is not, you must specify the user ID each time you set monitoring or a monitoring option. If you are using the Source Listing facility or the menus to set monitoring, press **PF1** to get help on how to monitor by user ID. If you are using CNTL commands, specify the USR option when you set monitoring, as follows:

```
CNTL=ON,promid,USR=userid
```

Monitor a Program at a Single Terminal

When a program fails in the production system, use CA InterTest for CICS to monitor it to find and correct the error. Restricting monitoring to a single terminal means you test the program thoroughly *without* interfering with the work of other users.

If you are using the CNTL Monitoring Command Builder menus, limit monitoring to a single terminal on the Replacement Options, Protection Options, and Special Options screens. If you are using commands, specify the TON option when you set monitoring, as follows:

```
CNTL=ON,promid,TON=termid
```

Monitor the Entire Production System

You usually want to avoid monitoring the entire production system because of the high overhead involved. But what if your production system has a serious error and you cannot isolate the problem program? In that case, instruct CA InterTest for CICS to monitor *all* the programs in the production system for every user or for a specific user only.

If your system is experiencing sporadic problems that are difficult to isolate, you might want to monitor all programs executed by the user experiencing the problems. In a secure CICS region, use the following global monitoring command with the user ID option for the user experiencing the problem:

```
CNTL=ON,PROG=.ALL,USR=cics-userid
```

Note: Ensure each user signs on to CICS before executing programs.

While CA InterTest is monitoring all of the programs, use the Active Tasks display (Primary Menu Option 2.5 Monitoring Active Tasks) to check for problems. When CA InterTest for CICS finds that a program is about to commit an error, it halts the program at an automatic breakpoint and automatically creates a Monitoring Table entry for that program. The monitoring status report provides that information. As soon as you see that CA InterTest for CICS has identified the problem program, turn monitoring off for the system and monitor the problem program for a single user or at a single terminal until you resolve the error.

Alternatively, monitor a certain percentage of your programs until CA InterTest for CICS finds the problem program.

Example

You might monitor 10 percent of your programs one week, another 10 percent the next week, and a third 10 percent the following week.

This approach minimizes overhead, and takes advantage of the ability of CA InterTest for CICS to track down problems that are otherwise very difficult to resolve.

Monitor Tasks That Execute Without Terminals

Special steps must be taken when CA InterTest for CICS monitors tasks that execute without a terminal. This includes tasks that are automatically initiated by an interval control request or a transient data trigger.

Unconditional, Conditional, and Request Breakpoints

If you set unconditional, conditional, and request breakpoints and you want to see the breakpoint display, you must indicate that the task is running without a terminal. Specify the terminal to receive breakpoint displays.

If you specify .ANY for the terminal at which the task must be executing, the breakpoint occurs at all terminals—even when the task runs without a terminal. If you specify .NO, the breakpoint occurs only when the task executes without a terminal.

Specify the ID of the terminal to receive breakpoint displays. If omitted, it defaults to the terminal at which the breakpoint was set. Do not specify .ANY for the terminal to receive breakpoint displays.

Note: If you set a breakpoint, the local automatic breakpoint option is automatically set for the program.

Automatic Breakpoints

If the global automatic breakpoint facility is on and a program is executing without a terminal, the task abends. To prevent this, route automatic breakpoint displays to 3270-type terminals. Use the global or local automatic breakpoint terminal options to identify the terminal to receive automatic breakpoint displays.

To identify a terminal to receive breakpoint displays for a specific program or transaction, select Special Options on the Monitoring Submenu for the Program (2.1) or Transaction (2.2). Specify a terminal ID for the local automatic breakpoint option on the Special Options menu. For details, see the chapter Monitoring Menu Options.

To identify a terminal to receive breakpoint displays for all non-terminal attached tasks and for tasks executing from non-3270 terminals, select System-Wide Options from the Monitoring Menu (Primary Menu Option 2.6, Breakpoint Menu Option 1.2.6). On the System-Wide menu, specify a terminal ID for the global automatic breakpoint option.

Set Monitoring for Non-Terminal Execution

Monitor a task only when it executes without a terminal. Specify this by selecting Replacement Options, Protection Options, or Special Options from Monitoring submenus (2.1, 2.2, or 2.3). Specify .NO in the Limit Monitoring field. You can also enter the following command:

```
CNTL=ON,promid,TON=.NO
```

For more information:

Monitoring Menu Options (see page 227)

Monitor Tasks That Execute at Non-3270 Terminals

Monitoring commands for tasks executing at non-3270 terminals must be entered at 3270-type terminals.

Set Monitoring for Non-Terminal Execution

Monitor a task only when it executes without a terminal. Specify this by selecting Replacement Options, Protection Options, or Special Options from the Monitoring submenus, and then specify .NO in the Limit Monitoring field. For details, see the chapter "Monitoring Menu Options." You can also enter this command:

```
CNTL=ON,promid,TON=.NO
```

For more information:

Monitoring Menu Options (see page 227)

Automatic Breakpoints

When a task is running on a non-3270 terminal, route automatic breakpoint displays to a 3270-type terminal. Use the global or local automatic breakpoint terminal options, described previously in Monitor Tasks That Execute Without Terminals (see page 272), to identify the terminal to receive automatic breakpoint displays.

Unconditional, Conditional, and Request Breakpoints

If you set unconditional, conditional, and request breakpoints and you want to get the breakpoint display, you must specify .ANY or a terminal ID for the terminal at which the task must be executing for breakpoints to take effect.

Route breakpoint displays to a specific terminal. To change the terminal ID: use the Source Listing Profile, and change the setting prior to setting any breakpoints.To see your current terminal's ID, enter **TERM** on the command line of any Source Listing or Breakpoint display.

If CA InterTest for CICS was installed with the DFLTUSER value in IN25OPTS set to SPECIFIC, Terminal ID =.ANY is the default terminal setting. When the DFLTUSER option is set to .ANY, the default terminal ID is the terminal used to set the breakpoint. To view your IN25OPTS settings, use Primary Menu Option 7.12.

For more information:

Source Listing Facility (see page 47)

Segmented Monitoring to Handle Special Situations

egmented monitoring is supported by CA InterTest releases 5.1 and higher, running on z/OS systems.

Users of the following might require segmented monitoring:

- HOGAN Systems
- DCI Interface of the MSA Corporation
- LE/370 condition handlers
- Some composite programs

CA InterTest for CICS normally controls whole application programs. Segmented monitoring provides a way for CA InterTest for CICS to control only small portions (code segments) of application programs.

Use segmented monitoring sparingly for special debugging needs, such as when only certain pieces of code should be monitored because of excessive overhead or because they get control directly from programs (or interfaces) that must not be monitored by CA InterTest for CICS. The following figure shows why segmented monitoring is used.

```
  CA InterTest for CICS r8.5   - DEMOSYM FILE  SOURCE LISTING DISPLAY
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 Profile   5 Monitor    6 Menu
     7 Backward     8 Forward    9            10           11           12 Status

--------------------------------------------------------------------------------
-
```

```
_ 000659     MOVE ALL '*' TO TASK-TEXT OF TASK-STRUCTURE-2.
_ 000660     GO TO SEND-MAP00.
_ 000661 INITIALIZE-TABLE.
_ 000662     MOVE SUB-1 TO STATE-NUMBER (SUB-1),
_ 000663                   COUNTY-NUM1 (SUB-1, SUB-2).
_ 000664     MOVE ','   TO COUNTY-COMM (SUB-1, SUB-2).
_ 000665     MOVE SUB-2 TO COUNTY-NUM2 (SUB-1, SUB-2).
  000666
_ 000667 DATA-NAME.
_ 000668     IF  TASK-SWITCH3 EQUAL SPACE
_ 000669     MOVE 'A' TO TASK-SWITCH3
```

Segmented monitoring ON and OFF options are used here:

- for a segment of code that needs testing or monitoring.
- and may not receive control in the normal way.

Important! Improper use of segmented monitoring options could damage your CICS system because CA InterTest for CICS places invalid machine instructions in application programs at the segmented monitoring locations specified by the user. Review Notes and Warnings (see page 286) before using segmented monitoring.

Segmented Monitoring Situations

For users running CA InterTest for CICS 5.*x* releases on a z/OS system, the monitoring problems that follow are solved with segmented monitoring.

HOGAN Systems Users

Problem:

CA InterTest for CICS must not monitor most of the HOGAN Systems code. However, you need to monitor and test the application code that you wrote. This code does not receive control from CICS in the usual way.

Solution:

Place a MON option to start CA InterTest for CICS monitoring at the beginning of your application program's Procedure Division. Monitoring begins at the MON location and continues until an activity #1 PEM call is encountered. The CA InterTest for CICS IN25UEX routine for HOGAN applications detects this call and stops the monitoring. Set breakpoints within this segment of code. When you execute the transaction, CA InterTest for CICS monitors the specified code segment.

You must also set the CA InterTest for CICS option USH=ON to prevent automatic breakpoints from being triggered when your HOGAN Application program references CICS shared-storage areas.

MSA Programs with Special Interfaces

Problem:

The application programs that hook into large packages such as MSA do not receive control directly from CICS, and thus cannot be monitored independently from the MSA packages using standard CA InterTest for CICS monitoring. Monitoring the entire load module incurs too much overhead and often results in numerous abends because of non-standard CICS programming practices (such as passing control using direct branches).

Solution:

Monitor only the application program that you wrote by setting a MON option near the beginning of the program and a NOM at the end of the program. Set breakpoints within this segment of code. When you execute the transaction, CA InterTest for CICS monitors only the specified code segment.

Note: For MSA programs set the MON after the secondary entry point.

The Segmented Monitoring Options Monitor (MON) and No Monitor (NOM)

Segmented monitoring consists of two independent monitoring options:

- MON, an option to turn monitoring on
- NOM, an option to turn monitoring off (No Monitor)

Monitor On Option (MON)

MON turns CA InterTest for CICS monitoring on beginning with the machine instruction of the specified location (regardless of how this piece of application program got control). *If CA InterTest for CICS monitoring was already on, monitoring continues.* After a MON entry, monitoring continues unless CA InterTest for CICS encounters one of the following:

- A NOM entry
- An XCTL
- A RETURN

No Monitor Option (NOM)

NOM stops CA InterTest for CICS monitoring at the specified location and gives control to the machine instruction on which the NOM option was specified. *If the code was not already being monitored, it continues unmonitored.* After a NOM entry, the program continues unmonitored unless CA InterTest for CICS encounters a MON entry or a LINK/XCTL to another program already being monitored.

Specify these options in your program at the same type of locations you use to set individual breakpoints, such as at COBOL paragraph names or statement numbers. Each option takes effect as it is encountered during program execution. Each MON and NOM option is an *independent* request to start or stop CA InterTest for CICS monitoring, regardless of the current monitoring status.

Note: The MON and NOM options are not CICS user ID or terminal sensitive. An option set from one terminal applies when the program executes from *any* terminal by any CICS user.

When using MON and NOM, specify breakpoints and other options in your program. If you set a MON or NOM option at *the same location* as a breakpoint or other option, the MON or NOM option executes first. This means the MON option starts monitoring and hits the breakpoint, but since NOM drops monitoring before the breakpoint is encountered, you never receive the breakpoint that is set where a NOM is also set.

Installation Requirements and Password Protection

The segmented monitoring option must be enabled as part of the installation and customization procedure. Enable it with or without CNTL menu support, and with or without password protection.

To check the status of the segmented monitoring option on your system, select Primary Menu Option 7 Status/Maintenance, and then Option 1.2 Global Options. The current IN25OPTS values are displayed alphabetically. Look for the option MONOM.

The following are MONOM values:

MENU—Segmented monitoring is enabled for all methods: Source Listing, CNTL menus, and CNTL commands.

NOMENU—Segmented monitoring is enabled for Source Listing and CNTL command methods only.

NO—Segmented monitoring is disabled.

The MONOMSEC keyword option tells you if a password is required. If MONOMSEC=YES, you need the password to set MON or NOM using any method.

Note: CA InterTest for CICS releases prior to 5.1 do not support segmented monitoring.

Methods for Setting and Removing MON and NOM

There are three methods for setting and removing MON and NOM, as shown following:

- Source listing
- CNTL menus
- CNTL one-line commands

The Source listing method is easiest to use.

Use Your Source Listing

This method requires a listing version that corresponds to your load module. If you use an old listing version, you can experience unpredictable results during execution—ranging from no effect, to unexpected abends, to data corruption.

Set MON and NOM

To set a location for a MON option in your source listing, enter a plus sign, +, to the left of the statement in your listing where you want monitoring to *begin*, and press Enter. Likewise, to set a location for a NOM option, enter a minus sign, -, to the left of the statement in your listing where you want monitoring to *stop*, and press Enter. If a password is required, enter the password at the prompt.

Example

The following screen shows setting MON and NOM in your source listing.

```
               CA InterTest  PROTDEM FILE SOURCE LISTING DISPLAY
COMMAND ===>
Program= COBDEMO  Option #       Stmt #                              Margin= 01
                  Label/Search Arg=
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           10           11           12 Status

   _ 01057     MOVE ALL '*' TO TASKTEXT OF TASKSTRUCTURE2.
   _ 01058     GO TO SENDMAP00.
➔+ 01059 INITIALIZETABLE.
   _ 01060     MOVE SUB1 TO STATENUMBER (SUB1),
   _ 01061                  COUNTYNUM1 (SUB1, SUB2).
   _ 01062     MOVE ','   TO COUNTYCOMM (SUB1, SUB2).
   _ 01063     MOVE SUB2 TO COUNTYNUM2 (SUB1, SUB2).
   _ 01064
➔- 01065 DATANAME.
   _ 01066     IF  TASKSWITCH3 EQUAL SPACE
   _ 01067     MOVE 'A' TO TASKSWITCH3
   _ 01068     MOVE 'DMAP02' TO MAPNAME
   _ 01069     GO TO SENDREWRITERETURN.
   _ 01070     GO TO SENDMAP00.
   _ 1071 SETVARREC.
```

Set breakpoints within the execution path of the MON and the NOM locations in the program.

Note: If you specify a MON or NOM option *and* a breakpoint at the same location, CA InterTest for CICS displays the character for the *first option entered* in column 1, although both options are set. To obtain a status display of the options in effect, enter **STATUS** in the command line, and press **Enter**.

Remove MON and NOM

To remove a MON or NOM option location, overtype the plus sign, +, or minus sign, -, with an X. Press **Enter**, and the option is removed.

To quickly remove all segmented monitoring options for a program, turn monitoring off for the program, remove the options from the Monitoring Status display, or use the **.**ALL entry on the CNTL Segmented Monitoring ON and OFF menus.

Access the specific CNTL menu directly from the Source Listing Display by making the following entries in the Option # field:

Option # Entry	CNTL Menu Function
=14S	Set segmented monitoring on

Option # Entry	CNTL Menu Function
=14R	Remove segmented monitoring on
=15S	Set segmented monitoring off
=15R	Remove segmented monitoring off

The CNTL Menus

When segmented monitoring is enabled with MENU support, set and remove segmented monitoring for programs directly from the following CNTL Function Selection Menu, using Options 14 (Segmented Monitoring On) and 15 (Segmented Monitoring Off).

```
                CA InterTest MONITORING COMMAND BUILDER  FUNCTION SELECTION


Select function by number: __ ❶
For functions 1022, enter S to set or R to remove: _ ❷

10  Monitoring                20  Replace Options    30  Status Display
11  Unconditional Breakpoints 21  Protect Options    31  Utility Functions
12  Conditional Breakpoints   22  Special Options    32  SystemWide Functions
13  Request Breakpoints       23  Composite Support  33  Resume Task Execution
14  Segmented Monitoring On   24  Indirect Commands
15  Segmented Monitoring Off
16  Statement Trace Options

For functions 1030, enter Program Names or Transaction Codes or Terminal Names

❸Program Names      COBDEMO  ________ ________ ________ ________ ________
   Transaction Codes  ____ ____ ____ ____ ____ ____ ____ ____ ____
   Terminal Names     ____ ____ ____ ____ ____ ____ ____ ____ ____

❹ user ID            ________  user ID (or .ANY) who will execute the program


1 Help       2           3 End       4 Return    5              6
```

- To set or remove a MON option, specify **14** in field ❶, S or R in field ❷, one program name in field ❸, and press Enter**.** To limit the monitoring option to a single user, also enter the user ID in field ❹ before pressing Enter**.**
- To set or remove a NOM option, specify 15 in field ❶, S or R in field ❷, one program name in field ❸, and press Enter. To limit the monitoring option to a single user, also enter the user ID in field ❹ before pressing Enter**.**

Notes:

- If Options 14 and 15 do not display on the Function Selection menu, menu support for segmented monitoring is not available to you.
- If segmented monitoring is password protected, you are prompted to enter the password.

- Specifying NOM options with the menus is tricky because of double negatives:
  - Use 15S to set a segmented monitoring off, or NOM, location.
  - Use 15R to remove a segmented monitoring off, or NOM, location that you previously set.

Specify segmented monitoring locations like you would specify individual breakpoint locations on the Segmented Monitoring On and Off screens.

CNTL Function 14S: Set Monitor On Screen

The following screen is used to set the monitor on locations in COBOL programs. The sample entry ❶ specifies that monitoring begins (or continues) whenever user CARAR01 executes the first executable statement in the paragraph A-TEST.

```
   CA InterTest MONITORING COMMAND BUILDER  SEGMENTED MONITORING LOCATIONS  14

  SET `monitor on' option for  PROG=COBDEMO in any of the following fields:
   (For qualification by nested program name, specify 'Programid:Label')
❶ Paragraph ATEST_____________________     _____________________________
     Names: _____________________________     _____________________________
            _____________________________     _____________________________
  Statement
   Numbers: _____    _____    _____    _____    _____   _____    _____    _____

   Offsets: _____    _____    _____    _____    _____   _____    _____    _____


            user ID (or .ANY) who will execute the program:            CARAR01
```

There are similar versions of this menu for Assembler and PL/I users. Specify multiple monitors on locations on this menu. Valid locations are as follows:

- COBOL (paragraph names, statement numbers, or hexadecimal offsets)

  Note: You cannot specify COBOL data names as segmented monitoring locations.
- Assembler (labels or hexadecimal offsets)
- PL/I (PL/I labels, statement numbers, or hexadecimal offsets)

Press **Enter** after specifying the locations.

CNTL Function 14R: Remove Monitor On Screen

Remove any option directly from the Monitoring Status display, as discussed in the chapter "Menus and Displays." Specify individual locations to be removed exactly as you set them or by the ID number shown on the CA InterTest for CICS status display. To quickly remove all locations at once, enter **.ALL** in the first field as shown on the menu ❶ and enter the same user ID (or .ANY) used to set the options.

The following screen is used to remove the monitor on locations that were previously set in the program for user ID CARAR01.

```
    CA InterTest MONITORING COMMAND BUILDER  SEGMENTED MONITORING LOCATIONS  14

    Remove `monitor on' option for  PROG=COBDEMO in any of the following fields:

   (For qualification by nested program name, specify 'Programid:Label')
❶Paragraph .ALL_____________________     _______________________________
    Names:  _______________________________     _______________________________
            _______________________________     _______________________________
  Statement
   Numbers: __    _____    _____    _____    _____   _____    _____    _____

   Offsets: _____    _____    _____    _____    _____   _____    _____    _____


           user ID (or .ANY) who will execute the program:             CARAR01

To remove by ID # shown in status display enter ID # s: -------------
```

After specifying the locations, press **Enter**.

CNTL Function 15S: Set Monitor Off Screen

The following screen is used to set locations for no monitoring in COBOL programs; it specifies two locations:

- Specifies the first executable statement in paragraph A-TEST-EXIT.
- Indicates statement number 125 in the program.

Program monitoring is dropped (or continues unmonitored) whenever these locations are executed.

```
  CA InterTest MONITORING COMMAND BUILDER  SEGMENTED MONITORING LOCATIONS  14

    SET `monitor off' option for  PROG=COBDEMO in any of the following fields:
   (For qualification by nested program name, specify 'Programid:Label')
❶Paragraph ATESTEXIT_____________________     _______________________________
    Names:  _______________________________     _______________________________
            _______________________________     _______________________________
  Statement  ❷
   Numbers: 125__    _____    _____    _____    _____   _____    _____    _____

   Offsets: _____    _____    _____    _____    _____   _____    _____    _____


           user ID (or .ANY) who will execute the program:             CARAR01
```

Note: There are similar versions of this menu for Assembler and PL/I users.

Specify multiple monitor off locations on this menu. Valid locations are as follows:

- COBOL (paragraph names, statement numbers, or hex offsets)

  Note: You cannot specify COBOL data names as segmented monitoring locations.
- Assembler (labels or hex offsets)
- PL/I (PL/I labels, statement numbers, or hex offsets)

After specifying the locations, press **Enter**.

CNTL Function 15R: Remove Monitor Off Screen

Specify individual locations to be removed exactly as you set them, or by the ID number shown on the CA InterTest for CICS status display. To quickly remove all locations at once, enter **.ALL** in the first field on the menu and the same user ID (or .ANY) used to set the options. The following screen is used to remove the monitor off locations that were previously set in the program using the ID number shown in the status report.

```
     CA InterTest MONITORING COMMAND BUILDER  SEGMENTED MONITORING LOCATIONS  14

   REMOVE `monitor off' option for  PROG=COBDEMO in any of the following fields:
   (For qualification by nested program name, specify 'Programid:Label')
 Paragraph ________________________________     ________________________________
    Names: ________________________________     ________________________________
           ________________________________     ________________________________
  Statement
   Numbers: __    _____    _____    _____    _____   _____    _____    _____

   Offsets: _____    _____    _____    _____    _____   _____    _____    _____


          user ID (or .ANY) who will execute the program:             CARAR01
To remove by ID # shown in status display enter ID #'s:   __ __ __ __ __ __ __
```

Press **Enter** after specifying the locations.

CNTL Commands

Review Notes and Warnings (see page 286) later in this chapter before using command syntax to specify segmented monitoring.

Set MON and NOM Options

To set one or more MON options; use either of the commands shown next. To set one or more NOM options; use NOM instead of MON in these commands.

```
CNTL=ON,PROG=progname,MON=locn
CNTL=ON,PROG=progname,MON=(locn1,locn2, ...,locn9)
```

progname

Indicates one program.

locn

Specifies an address (from six- to eight-hexadecimal digits), offset (from one- to five-hexadecimal digits), statement number (a decimal number preceded by #), COBOL paragraph name, Assembler or PL/I label (enclosed in apostrophes).

To specify a location, use the same format as for breakpoints.

At a breakpoint display, enter the abbreviated command form to see this Monitor Table entry:

```
CNTL=ON*,MON=... or  CNTL=ON*,NOM=...
```

Other options, including breakpoints, precede or follow within the same CNTL command.

Remove MON and NOM Options

To remove one or more MON options, use one of the following command forms. To remove all NOM options, use NOM instead of MON in these commands

```
CNTL=OFF,PROG=progname,MON=ID=id
CNTL=OFF,PROG=progname,MON=locn
CNTL=OFF,PROG=progname,MON=ID=(id1,id2, ...,id9)
CNTL=OFF,PROG=progname,MON=(locn1,locn2, ...,locn9)
```

progname

Indicates the name of one program.

id

Indicates same number as the ID= number displayed by the CA InterTest for CICS Inquiry Report (obtained by CNTL=INQ or, at breakpoint, by pressing PF3).

locn

Specifies an address (from - to eight-hexadecimal digits), offset (from one to five hexadecimal digits), statement number (a decimal number preceded by #), COBOL paragraph name, Assembler or PL/I label (enclosed in apostrophes).

To specify a location, use the same format as for breakpoints.

Specify up to nine IDs or locations, separated by commas and enclosed in parentheses.

To remove all MON options for a Monitor Table entry, use the form:

```
CNTL=OFF,PROG=progname,MON=.ALL
```

At a breakpoint, use the abbreviated command form to see this Monitor Table entry:

```
CNTL=OFF*,MON=...
CNTL=OFF*,NOM=...
```

Other options, including breakpoints, precede or follow within the same CNTL command.

Note: The command CNTL=OFF,PROG=*progname* does not remove the MON or NOM entries for a program. To remove the MON and NOM options, you must explicitly specify them.

Examples

The following are command examples of specifying MON and NOM options.

To selectively monitor the program COBPROGA from paragraph A-TEST to the paragraph name A-TEST-EXIT, specify:

```
CNTL=ON,PROG=COBPROGA,MON='A-TEST'
```

This command declares a MON option for the first machine instruction in A-TEST. CA InterTest for CICS monitoring starts when the first machine instruction at A-TEST is executed.

The following command declares a NOM option for the first machine instruction in statement 123. CA InterTest for CICS monitoring stops prior to the execution of this statement.

```
CNTL=ON,PROG=COBPROGA,NOM=#123
```

The following command declares a NOM option for the first machine instruction in the paragraph A-TEST-EXIT.

```
CNTL=ON,PROG=COBPROGA,NOM='A-TEST-EXIT'
```

The following command declares the same options as the three preceding commands.

```
CNTL=ON,PROG=COBPROGA,MON='A-TEST',NOM=(#123,'A-TEST-EXIT')
```

The following command selectively removes the two NOM options that were displayed by an Inquiry Report with ID= numbers 02 and 03.

```
CNTL=OFF,PROG=COBPROGA,NOM=ID=(02,03)
```

```
CNTL=OFF,PROG=COBPROGA,MON=.ALL,NOM=.ALL,LET=.ALL
```

Note: Any non-sequential passing of control (such as PERFORM, GO TO, and so on), executed between the MON location at A-TEST and the NOM location at A-TEST-EXIT could result in CA InterTest for CICS monitoring outside the designated range.

Notes and Warnings

The following information contains notes and warning that you should be aware of when using CA InterTest for CICS.

Location Restrictions

The following bullets list location restrictions:

- Regardless of how they are set (Source Listing, Menu, or single-line command); each of these options must be placed on the first byte (the op-code) of a machine instruction that resides in non-store-protected virtual storage. Because the code must not be store-protected, the application programs must not reside in the LPA, ELPA, or ERDSA.
- Regardless of how they are set (Source Listing, Menu, or single-line command); each of these options must be placed on the first byte (the op-code) of a machine instruction that resides in non-store-protected virtual storage. Because the code must not be store-protected, the application programs must not reside in the LPA, ELPA, or ERDSA.
- Any piece of code with these options must remain resident at the same virtual storage address until the options are removed.
- These options cannot be placed on sequential statement numbers or instruction addresses; that is, these options must be separated by at least one machine instruction.
- In Command Level CICS applications within code generated by an EXEC CICS, these options must not be placed on the LOAD 15 that precedes the BALR 14,15 or on that BALR.

Usage Notes and Restrictions

The following bullets list usage notes and restrictions:

- Declare other CA InterTest for CICS options, such as breakpoints, bypasses, and so on, at the same locations as the MON and NOM options
- MON and NOM options are not remembered by the CA InterTest for CICS checkpoint, nor are they restored by the CA InterTest for CICS restart (CNTL=RESTART).
- The CA InterTest for CICS New Copy function does not work until all MON and NOM options are removed from the affected program. The CEMT SET PROG (xxxxxxxx) NEWCOPY must not be performed until all MON and NOM options have been removed.
- There is a limit of 255 MON or NOM options per CICS region.
- A MON option causes CA InterTest for CICS monitoring, regardless of the CA InterTest Exclusion Table.
- Use these options to control application programs only. CA InterTest for CICS will accept them outside the scope of applications programming, such as CICS Exits or subsystem interfaces, but the results are unpredictable.
- Do not place MON entries in programs that are used in PLT shutdown processing.
- CA InterTest for CICS monitoring for a program cannot be removed until any MON and NOM options declared for the program are removed. To remove all MON and NOM options, specify:

  ```
  CNTL=OFF,PROG=progname,MON=.ALL,NOM=.ALL
  ```

Programmed Breakpoints Usage

Programmed breakpoints (PBP) are intended primarily for users of PL/I who want to display data areas that CA InterTest for CICS normally cannot display or who do not have symbolic information for their programs. Incorporate programmed breakpoints into COBOL and Assembler code.

Programmed breakpoints differ from other CA InterTest for CICS breakpoints because they are actually coded by the programmer into the tested program. A programmed breakpoint occurs because it was written into the monitored program in the form of a special CALL statement that identifies the data areas to display. The Detailed Breakpoint display automatically shows, in dump format, up to three data areas named in the CALL statement parameters. There is no need to enter CORE commands to display these areas. However, if there are more than three data areas, use the command CORE=ARGnn to display them.

If you are using the Source Listing Breakpoint facility, CA InterTest for CICS stops the program at the programmed breakpoint. Press PF3 DET BKPT to display the Detailed Breakpoint screen to view the requested data areas.

The programmed breakpoint will not occur unless the automatic breakpoint facility is active. If this facility is not active, the programmed breakpoint CALL has *no effect* on program execution. The programmed breakpoint also does not affect program execution if the program is not monitored by CA InterTest for CICS (the called routine only does a return). However, a small amount of overhead is added to the program's execution. For this reason, programmed breakpoints should be removed before the program is put into production.

Programmed Breakpoints Coding

To execute a programmed breakpoint, add a special CALL statement to the program source code at the location where you want the breakpoint to occur. The CALL statement passes control to an entry point named PBP. Code the CALL statement according to the rules of the programming language being used. It also passes a number of arguments or parameters. Include a literal that describes the location of the programmed breakpoint as the first parameter so that the user, especially the PL/I user, identifies the breakpoint easily.

When control comes to the program location where the programmed breakpoint should occur, CA InterTest for CICS executes the programmed breakpoint just like an automatic breakpoint, except that in the data portion of the display 16 bytes of the first three passed parameters are automatically displayed. In addition, a CORE command keyword is provided to display the parameters.

The small Assembler programmed breakpoint load module, which is identical for all programming languages, must be prepared by your systems programmer and assembled and link-edited into the proper language library so that it is automatically included in your module as a result of the CALL. The program follows:

```
PBP     CSECT
        SR      15,15
        BR      14
        DC      C'PBPINTERTEST'
        END     PBP
```

Declare a Programmed Breakpoint in a PL/I Program

A PL/I program with a programmed breakpoint must be coded to define the entry point as passing parameters in Assembler fashion, as shown next:

```
DCL PBP ENTRY OPTIONS(ASSEMBLER);
```

Once this is done, code the calls in the usual manner.

Example

```
WRITE_RESPONSE:
  RESPONSE = ERROR CODE;
  CALL PBP ('WRITE_RESPONSE',RESPONSE,ADDR(REC_SIZE));
```

At this programmed breakpoint, the breakpoint display shows the following data areas:

- The characters WRITE_RESPONSE tell you which programmed breakpoint (if many was coded) is being executed
- The first 16 bytes of the field named RESPONSE
- The full word that contains the address of the field named REC_SIZE

Note: If you do not include parameters in the CALL statement, a programmed breakpoint still occurs but no data is displayed.

Declare a Programmed Breakpoint in a COBOL Program

The CALL statement that sets up a programmed breakpoint in a COBOL program is coded in the standard manner as follows:

```
WRITE-RESPONSE.
  MOVE ERROR-CODE TO RESPONSE.
  CALL 'PBP' USING RESPONSE.
```

At this programmed breakpoint, the breakpoint display shows the first 16 bytes of the field named RESPONSE.

If you do not include parameters in the CALL statement, a programmed breakpoint still occurs but data is not automatically displayed.

Declare a Programmed Breakpoint in an Assembler Program

In Assembler you must use the CALL macro, coded so that the last parameter is marked as such, or coded without parameters. Because of the restrictions of the CALL macro, it is often practical not to code parameters. The breakpoint display shows the location of the programmed breakpoint and the contents of the registers.

If you do not include parameters in the CALL statement, a programmed breakpoint still occurs but data is not automatically displayed.

Remove a Programmed Breakpoint

Once you code a programmed breakpoint CALL in a program, the breakpoint occurs every time execution reaches the programmed breakpoint's location during an interactive session. The only way to deactivate the programmed breakpoint is to replace the BALR 14,15 instruction in the program, which is contained in the CALL statement, by a NOPR 0. When the task is waiting at breakpoint, do this easily by issuing the command:

```
CORE=CURR=CHGX'0700'
```

However, since this modifies the program's code, the programmed breakpoint will not occur again until the program is reloaded by CICS.

After program testing is complete, the programmed breakpoint CALL statements should be removed for improved performance.

Monitoring CICS/ESA 3.3 FEPI Applications

The Front-End Programming Interface (FEPI) provides terminal emulation using virtual terminals defined to a CICS region. These virtual terminals, called nodes, communicate with back-end processes that run either in the same CICS region or in a remote region. Front-end processes, which are coded with EXEC CICS FEPI commands, communicate with both real and virtual terminals.

CA InterTest for CICS releases 5.2 and higher let both front-end and back-end processes be tested. However, the terminal IDs must be properly specified when breakpoints are set.

Set Monitoring and Breakpoints for a FrontEnd Process

Set monitoring and breakpoints as usual when testing a front-end process only.

Set Monitoring and Breakpoints for a BackEnd Process

Turn on monitoring as usual for a back-end process running in a local CICS region.

To set breakpoints for a back-end process running in a local region, specify both the terminal at which the process is running and the terminal to receive breakpoints, as follows:

- Term ID where breakpoints take effect: (F=)

  The virtual terminal assigned by CICS. Specify either an actual virtual terminal name or .ANY if the terminal is assigned from a pool and its name is not known.

- Term ID that receives the breakpoints: (T=)

  The real terminal to which breakpoints should be sent.

It is important that breakpoints **not** be sent to a virtual terminal because this could disrupt the application dialog screen flow.

If the back-end process is running in a remote CICS region, turn on monitoring in the remote region and ensure that the terminal that receives breakpoints is a real terminal connected to the remote region.

Set Breakpoints for Both FrontEnd and BackEnd Processes

To set breakpoints for both front- and back-end processes that are running concurrently, follow the steps outlined in the previous sections. However, make sure that *two real terminals* are available for receiving breakpoint displays: one for breakpoints from the front-end process, and one for breakpoints from the back-end process. For example, if both processes are running in the same CICS region, you might enter the following commands from TER1:

```
CNTL=ON,PROG=FRONTEND,UBP=#1
CNTL=ON,PROG=BACKEND,UBP=((#1),F=VIR1,T=TER2)
```

Breakpoint displays for program FRONTEND are sent to TER1, the terminal at which the command was issued. Breakpoint displays for program BACKEND are sent to TER2.

If the back-end process is running in a remote CICS region, perform these steps:

1. Set monitoring and breakpoints for the *front-end* process in the local region.
2. Set monitoring and breakpoints for the *back-end* process in the remote region from any real terminal.

TIMEOUT Values

If a TIMEOUT(seconds) parameter is specified on an EXEC CICS FEPI command when CA InterTest for CICS is monitoring a back-end process, all breakpoint activity must be completed within the specified time. Otherwise, the front-end virtual terminal session times out and the back-end process could abend. To avoid this, it is strongly recommended that the FEPI TIMEOUT value be set to *zero* during testing. Change it to an appropriate value when testing is completed.

CA InterTest for CICS with IBM's EDF (EXEC Debugging Facility)

The EDF facility, which is activated by the CEDF transaction and described in the IBM *CICS/VS Application Programmer's Guide Command Level*, does not interfere with EDF, except when EDF presents the EDF breakpoint display of the program being monitored by CA InterTest for CICS. EDF incorrectly indicates the location of the command since commands are issued by CA InterTest for CICS, not by the program. Additionally, CEDF is not supported on transactions that contain programs that use COMET.

Advantages of CA InterTest for CICS Over EDF

CA InterTest for CICS offers many advantages over the EDF facility, including the following benefits:

- The ability to set breakpoints anywhere in the program (not just at EXEC CICS commands).
- Data display/modification by symbolic names. This means the programmer does not need the most recent listing of the tested program, as long as the data names or paragraph names (labels in Assembler) remain the same.
- Monitoring with the ability to detect any illegal action of the program between CICS commands. CICS abends are intercepted by EDF, but damage might have occurred on the way to the abend, and EDF might not give any specifics on the problem.
- The ability to declare an unconditional or conditional breakpoint at a specific location—the EXEC CICS call at the point when the EXEC CICS call parameters are already formatted. At that time change the parameters (for example, by issuing the CORE=ARGnn command) before you let the command execute.
- The ability to set request breakpoints for a particular type of CICS command, regardless of where in the program it occurs, or for all EXEC CICS commands, or for all except some EXEC CICS commands. For example, with one specification set breakpoints at all File Control commands or at all READ or WRITE commands.

Using CA InterTest for CICS with EDF

If you want to use CA InterTest for CICS with EDF, turn on EDF before you begin monitoring a program with CA InterTest for CICS. Moreover, when a monitored program is stopped at a request breakpoint for an EXEC CICS command, activate EDF by entering any character in the field marked EDF in the lower right corner of the Detailed Breakpoint display.

When CA InterTest for CICS and EDF are being used on the same task, be aware of the following points:

- If a CA InterTest for CICS breakpoint is set at an EXEC CICS command, the CA InterTest for CICS breakpoint occurs before the command is passed to CICS.
- At the breakpoint, the programmer reviews and changes any parameters of the command before telling CA InterTest for CICS to continue with the task.
- CA InterTest for CICS checks the parameters and, if necessary, halts the task at an automatic breakpoint.
- If that does not happen, the command is passed by CA InterTest for CICS to CICS for execution; that is, to the EXEC Interface.

- Subsequently, EDF presents the before a command breakpoint. The EDF facility, however, is in control at that time and any changes by you during the EDF breakpoint cannot be checked by CA InterTest for CICS.
- Only after the command is executed, the EDF breakpoint display opens, and EDF is told to continue with the task can CA InterTest for CICS resume control and continue monitoring.

Chapter 9: CNTL Commands and Menus

The primary task of CA InterTest for CICS is to monitor programs so that it detects and prevents errors. The CA InterTest for CICS CNTL transaction performs all monitoring functions.

Note: The CNTL Monitoring Command Builder menu topics are included in this chapter as a reference for CA InterTest for CICS users familiar with this method of performing monitoring functions from previous releases.

For more information:

Monitoring Menu Options (see page 227)

CNTL Monitoring Commands and Options

Enter single-line commands instead of using menus and screens. Entering commands directly is often faster and more efficient for experienced users. The CA InterTest for CICS CNTL transaction performs all monitoring functions. This topic teaches you which CNTL single-line commands you use to specify monitoring, monitoring options, and system-wide options. For example, to set monitoring for the program COBDEMO when executed by the user ID BARNEY1, and also set a breakpoint at the beginning of the program, enter the following single command from CICS:

```
CNTL=ON,PROG=COBDEMO,USR=BARNEY1,UBP=#1
```

CNTL Commands at a Glance

The following table indicates the functions you specify with commands and, where appropriate, the equivalent selections on the Primary Option menu, Breakpoint menu, and CNTL Function Selection Menu.

CNTL Command	Description	Menu Path	CNTL Function
START, END	Initializes and terminates CA InterTest for CICS.	Not available	Not available.
RESTART	Restarts CA InterTest for CICS.	Not available	Not available.
ON, OFF	Sets and removes monitoring.	Primary 2.1, 2.2, 2.3	10

CNTL Command	Description	Menu Path	CNTL Function
ON, OFF	Sets and removes monitoring options.	Primary 2.1, 2.2, 2.3	11 to 23
OFFALL	Removes ALL monitoring options for ALL programs/transactions/terminals set by a Userid or Terminal	Not available	Not available
EXCL, INCL	Excludes programs from monitoring and ends program exclusion; applies to all CICS user IDs.	Not available	Not available.
GO	Resumes task execution.	Breakpoint 4	33
GO	Abends a task.	Breakpoint 3	33
INQ, LIST	Produces CA InterTest for CICS reports.	Type STATUS	30 and 31
NEW	Loads a new copy of a program.	Primary 2.1, 2.2, 2.3	31
ABP, GLOG, CKPT,PURGE	Sets system-wide options.	Primary 2.6	32
EXEC	Executes a module of CNTL commands.	Not available	Not available.

Note: For more information about specifying CNTL Monitoring Options, CNTL Monitoring Options (see page 311) later in this chapter.

CNTL Command propagation in a CICSPlex Environment

For CICSPlex environments using the CA InterTest for CICS CICSPLEX=YES installation option, all CNTL functions that are checkpointed are propagated to all CICSPlex family members on startup (CNTL=START) or restart (CNTL=RESTART). The following table summarizes which CNTL commands are propagated and which are not.

Commands Propagated to All Family Members	Commands Not Propagated	Comments
CNTL=ABP		
CNTL=ADD		
CNTL=CKPT		
CNTL=DUMP		

Commands Propagated to All Family Members	Commands Not Propagated	Comments
CNTL=EDIT		
CNTL=END, {SCOPE=GLOBAL}		Propagated if SCOPE=GLOBAL is specified.
CNTL=EXCL		
CNTL=FOL		
CNTL=GLOG		
CNTL=INCL		
CNTL=ITTRACE		
CNTL=NEW		
CNTL=OFF		
CNTL=ON		
CNTL=OFF,ALL		
	CNTL=EXEC	Commands in the module are propagated if they are eligible for propagation.
	CNTL=GO	
	CNTL=INQ	
	CNTL=LIST	
	CNTL=MENU	
	CNTL=PURGE	
	CNTL=VRPT	

CNTL Command Syntax

CNTL command syntax uses the following notational conventions:

- Uppercase words (such as CNTL) are CA InterTest for CICS keywords that must be entered exactly as shown
- Lowercase words (such as promid) represent user-supplied information and should be replaced with the appropriate entries.
- Information in brackets is optional and can be omitted

The following list contains the terms used in command syntax for which you must substitute the appropriate information.

promid

Identifies one or more programs, transactions, or terminals. Use one of the following formats:

```
PROG=program
PROG=(prog1,...,prog9)
TRAN=trancode
TRAN=(tran1,...,tran9)
TERM=terminal
TERM=(term1,...,term9)
PROG=.ALL                to specify the entire system
```

Note: * replaces promid for the monitoring table entry currently at a breakpoint at your terminal.

progname

Identifies a program.

locn

Identifies a program location. Specify a location in one of the following ways:

- One- to five-digit hexadecimal displacement (offset) from the beginning of the program
- Six- or eight-digit hexadecimal address
- A COBOL paragraph name enclosed in single quotes
- A COBOL or PL/I statement number preceded by a # sign
- A PL/I or Assembler label enclosed in single quotes

options

Indicates one or more CNTL monitoring options. Separate multiple options with commas.

Enter CNTL commands directly on a clear screen or on the command line of a source listing, breakpoint, or CORE display.

Generic Specification of promid

Specify asterisks, *, and plus signs, +, to replace characters in *promid* in the following CNTL commands:

```
CNTL=ON,...
CNTL=OFF,...
CNTL=EXCL,...
CNTL=INCL,...
CNTL=INQ,...
```

Note: An asterisk, *, specifies any number of characters, including no characters.

These specifications are interpreted according to the rules for the CICS CEMT transaction.

Examples

To specify all programs that start with ABC, including program name ABC, enter:

```
PROG=ABC*
```

To specify all programs that end with ABC, including program name ABC, enter:

```
PROG=*ABC
```

To specify all programs that start with AB and end with C and all those that could have an unspecified number of characters in the middle, enter:

```
PROG=AB*C
```

Note: Specifies more than one asterisk in a promid. Do not specify an asterisk by itself: PROG=* because it is invalid.

To specify all four-character program names that begin with ABC followed by any one character, enter:

```
PROG=ABC+
```

Note: A plus sign, +, specifies only one character in a particular position.

To specify all four-character program names that begin with any character followed by ABC, enter:

```
PROG=+ABC
```

To specify all four-character program names that begin with AB followed by any one character and C, enter:

```
PROG=AB+C
```

Note: Specify more than one plus sign in a promid. Do not specify a plus sign by itself: PROG=+ because it is invalid.

To specify all programs that begin with AB followed by one character, C, and an unspecified number of characters, enter:

```
PROG=AB+C*
```

Note: Combine asterisks and plus signs.

Hierarchy Rules for Monitored Entries

It is important to know the priorities CA InterTest for CICS observes if more than one entry in the monitoring table applies to the same program.

- The PROG=.OPTIONS specification overrides all other specifications
- A program entry (specific or generic) overrides a transaction entry, a terminal entry, and an .ALL specification
- A transaction entry (specific or generic) overrides a terminal entry and an .ALL specification
- A terminal entry (specific or generic) overrides an .ALL specification
- A specific user ID overrides an .ANY specification

Example

Suppose you specify monitoring options for program ABC1 and different options for transaction TRN1. ABC1 executes as part of that transaction. When ABC1 executes, the options specified for ABC1 override the options specified for TRN1.

Similarly, it is important to understand the priorities CA InterTest for CICS observes when a program, transaction, or terminal is specifically and generically declared for monitoring. In this case, a specific entry overrides a generic entry except when you issue the command CNTL=INQ,promid, where the generic specification takes precedence.

Example

Suppose you specify monitoring options for program ABC1 and different monitoring options for program ABC*. The options specified for ABC* affect all programs whose names begin with ABC except for program ABC1 (and any other programs beginning with ABC that have their own entries in the monitoring table). Monitoring for ABC1 is controlled by the specific entry for that program. However, if you specify CNTL=INQ,PROG=ABC*, you get monitoring status reports for all programs that begin with ABC, including programs such as ABC1, which have separate monitoring table entries.

Initialize and Terminate CA InterTest for CICS

To initialize CA InterTest for CICS, enter:

```
CNTL=START
```

To terminate CA InterTest for CICS, enter:

```
CNTL=END
```

Important! This command terminates CA InterTest for CICS for all users and could be password protected.

Restart CA InterTest for CICS

To restart CA InterTest for CICS, enter:

```
CNTL=RESTART
```

or

```
CNTL=RESTART,TODAY=ONLY
```

This command restarts CA InterTest for CICS from the last recorded checkpoint. If TODAY=ONLY is specified, CA InterTest for CICS is restarted only if the checkpoint occurred on the same day.

When CA InterTest for CICS is restarted from a sequential terminal, it ignores all input records following the RESTART command until it finds a record with this command:

```
CNTL=DUMMY
```

This terminates the CNTL transaction and permits the system to execute transactions from subsequent input records. However, if the restart is unsuccessful, the CNTL transaction terminates and all input records that follow the RESTART command are executed as usual by CICS.

Set and Remove Monitoring

To set monitoring, enter:

```
CNTL=ON,promid
```

To set monitoring for all programs and all users, enter:

```
CNTL=ON,PROG=.ALL,USR=.ANY
```

To set monitoring for all programs executed by the user whose CICS user ID is BARNEY1, enter:

```
CNTL=ON,PROG=.ALL,USR=BARNEY1
```

To remove monitoring, enter:

```
CNTL=OFF,promid
```

Examples

To turn on monitoring for program COBDEMO enter:

```
CNTL=ON,PROG=cobdemo
```

To turn off monitoring for programs COBDEMO and PAYPROG, enter:

```
CNTL=OFF,PROG=(cobdemo,payprog)
```

Automatic Creation of Monitoring Table Entries

If you set monitoring for a transaction or terminal and a program executing as part of that transaction or at that terminal is stopped at an automatic breakpoint, CA InterTest automatically creates an entry in the monitoring table for that program. This enables you to set breakpoints and monitoring options for that individual program. Similarly, if you specified CNTL=ON,PROG=.ALL or you specified a generic program name, CA InterTest for CICS creates an individual monitoring table entry for any program monitored under that entry that CA InterTest for CICS stops at an automatic breakpoint.

Set and Remove Monitoring Options

To set monitoring options, enter:

```
CNTL=ON,promid,options  or  CNTL=ON*,options
```

To set system-wide default monitoring options, enter:

```
CNTL=ON,PROG=.OPTIONS,USR=.ANY,options
```

Options specified on a system-wide basis take effect for all programs CA InterTest for CICS monitors.

Notes: Specify only the following options on a system-wide basis: ABI, CSA, FOL, ICT, LET, MXR, MXS, NUP, PRO, RFC, RNT, RPC, RTD, RTS, SLB, STR, and TRC.

To remove monitoring options, enter:

```
CNTL=OFF,promid,options OR CNTL=OFF*,options
```

Examples

To set the Follow option for the task currently at a breakpoint at your terminal, enter:

```
CNTL=ON*,FOL=ON
```

To set the No Update option for all monitored tasks, enter:

```
CNTL=ON,PROG=.OPTIONS,NUP=ON
```

To remove the unconditional breakpoint at statement number 1 for program COBDEMO, enter:

```
CNTL=OFF,PROG=cobdemo,UBP=#1
```

Set and Remove ALL Monitoring Options Set by a User ID or Terminal

To remove ALL monitoring options set by a specific User ID or for a specific Terminal, enter:

```
CNTL=OFF,ALL,USR=userid or CNTL=OFF,ALL,TTR=termid,FTR=termid
```

TTR= and FTR= must specify the same terminal ID.

Use this command with *caution* because it removes monitoring options for more than one program or transaction. This command is also passed to the CNTL transaction as data on a CICS START command and could be used by an installation's AutoInstall exit when a terminal signs off to *automatically* remove all CA InterTest for CICS breakpoints set by the terminal or user ID.

Exclude Programs from Monitoring

To exclude a program from monitoring, enter:

CNTL=EXCL,promid

To remove the monitoring exclusion set by the CNTL=EXCL command, enter:

CNTL=INCL,promid

Note: INCL and EXCL are not sensitive to the monitoring user ID.

Examples

To exclude all of transaction PYRLs programs from monitoring, enter:

CNTL=EXCL,TRAN=PYRL

To remove program PAYROLL1 from the exclusion list so it is again eligible for

monitoring, enter:

CNTL=INCL,PROG=payroll1

Resume Task Execution

To resume task execution at a breakpoint, enter:

CNTL=GO,TASK=taskid,C[,options]

taskid

Represents the number of the task at a breakpoint. Use the CNTL=INQ command to determine the *taskid*.

C instructs CA InterTest for CICS to continue execution.

To resume execution of the task currently at a breakpoint at your terminal, enter:

CNTL*C[,options]

To resume task execution after taking a transaction dump, enter:

CNTL=GO,TASK=taskid,Ddddd[,options]

D*dddd, dddd*

Specifies a four-character transaction dump code.

Specify the following options with the CNTL commands for resuming execution:

```
AT=locn
```

or

```
AT=X
GO=locn
NOBREAK
S=#nnn[,I=nn][,N=CALL/xxxx]orS=nnn[,I=nn][,N=CALL/xxxx]
```

AT=*locn* moves the current breakpoint to a different location.

AT=*X* disables the breakpoint.

GO=*locn* specifies the location at which the task should resume execution. For details, see Limitations of Using the GO=Element.

NOBREAK resumes execution from the current statement ignoring all preset breakpoints and indirect commands until the task either abends or runs to normal completion. Use the NOBREAK option on:

```
CNTL*C
```

or

```
CNTL=GO, TASK=xxxxx, C
```

S=#nnn[,I=nn][,N=CALL/xxxx] instructs CA InterTest for CICS to halt program execution after *nnn* COBOL verbs or PL/I statements are executed.

S=*nnn*[,I=*nn*][,N=CALL/*xxxx*] instructs CA InterTest for CICS to halt program execution after nnn machine instructions are executed.

- I=*nn* instructs CA InterTest for CICS to repeat the single-stepping after pausing for *nn* seconds
- N=CALL halts automatic single-stepping at a CALL or CICS command or macro
- N=*xxxx* halts automatic single-stepping after a specified number (1-9999) of steps

Note: S=1 is specified as S.

Examples

To specify that task 12345 resume at COBOL paragraph name TASK-NUMBER-FOR-MSG, enter:

```
CNTL=GO,TASK=12345,C,GO=`task-number-for-msg'
```

To specify that the task currently at a breakpoint at your terminal resume at statement number 142 and that the program execute five COBOL verbs or PL/I statements before being halted, enter:

```
CNTL*C,GO=#142,S=#5
```

To specify that task 12345 resume execution after taking a dump and that the current breakpoint be disabled, enter:

```
CNTL=GO,TASK=12345,Dhelp,AT=X
```

The transaction dump code is HELP.

To specify that task 777 resume execution at indirect command sequence number 10, enter:

```
CNTL=GO,TASK=777,C,EXEC=0010
```

GO= Element Limitation

When changing the execution sequence of a high level program, especially COBOL or PL/I, you must take responsibility for securing proper *addressability* of the data that will be processed in the piece of logic specified by GO= element.

In COBOL, before you force a branch (GO TO) to the routine using GO= element, the BLLs and their corresponding registers must contain correct addresses. Accomplish this with CORE commands, especially the =SET element of the CORE command. Use symbolic names of BLL cells. The BLL cells are related to their data structures in the Data Division Map (DMAP) of the COBOL listing, while the related registers are found in the register assignment section of the memory map.

SERVICE RELOAD statements in the COBOL program are necessary to notify the COBOL compiler that, at a specific time, the registers must be reloaded when the address of the base of a certain data structure changes. If you force a branch (GO TO) to the intended routine using the GO= element and, in the process, you cross the boundaries of SERVICE RELOAD, then it is necessary to set the addresses.

In PL/I, you not only must secure the proper addressability of data, but also remain within the same logical level (such as the same DO block or called procedure). Going to another logical level requires going through all intermediate epilogues and prologues.

Abend a Task Stopped at a Breakpoint

To abend a task *with* or *without* a transaction dump, enter:

CNTL=GO,TASK=taskid,Adddd[,options]

taskid

Represents the number of the task at a breakpoint. Use the CNTL=INQ command to determine the taskid.

Adddd dddd

Specifies a four-character transaction dump code. Replace **dddd** with **XXXX** to abend the task *without* a dump and disable exits.

Note: A dump code of XXXX is not allowed for tasks in a MRO session. Specify A by itself to abend the task *without* a dump and enable the exits.

Specify the following options:

`AT=locn  or  AT=X`

AT=*locn* moves the breakpoint to a different location

AT=X disables the breakpoint

Examples

To abend task number 54321 with a dump and a transaction dump code of DMP1, enter:

CNTL=GO,TASK=54321,Admp1

To abend task number 54321 without a dump, enter:

CNTL=GO,TASK=54321,AXXXX

To abend the current task with a dump and move the current breakpoint to statement number 128, enter:

CNTL*Admp1,AT=#128

To abend the current task without a dump, enter:

CNTL*AXXXX

How to Produce CA InterTest for CICS Reports

To produce a monitoring status report, enter:

```
CNTL=INQ,promid[,TODEST=destid]
```

To produce a monitoring status report for the task currently at a breakpoint at your terminal, enter:

```
CNTL?
```

To produce a system-wide monitoring report, enter:

```
CNTL=INQ[,TODEST=destid]
```

To display the contents of CA InterTest for CICS monitor tables, enter:

```
CNTL=LIST[,TODEST=destid]
```

To display the entire contents of the CA InterTest for CICS symbolic file, enter:

```
CNTL=INQ,SYM=ALL[,TODEST=destid]
```

To display information for one program in the CA InterTest for CICS symbolic file, enter:

```
CNTL=INQ,SYM=progname[,TODEST=destid]
```

destid

Represents the transient data destination. If omitted, the report displays on your terminal screen.

Examples

To request a monitoring status report that displays on your terminal screen for program COBDEMO, enter:

```
CNTL=INQ,PROG=cobdemo
```

To request a monitoring status report for all programs that is routed to the CSSL transient data queue, enter:

```
CNTL=INQ,TODEST=CSSL
```

To request a report listing information on all programs in the symbolic file that is routed to the CSSL transient data queue, enter:

```
CNTL=INQ,SYM=ALL,TODEST=CSSL
```

Load a New Copy of a Program

To load a new copy of a program, reset symbolic breakpoints, and reset the entry in the PPT to the program's new library address, enter:

```
CNTL=NEW,PROG=progname
```

Note: The specified program is declared for monitoring before any breakpoints are reset. Monitoring and breakpoints are reset for all user IDs.

Example

To load a new copy of program COBDEMO.

```
CNTL=NEW,PROG=COBDEMO
```

Specify optionally a SYM=.NOASK parameter. Doing so causes InterTest Symbolic File Date Checking module to bypass ASKing the terminal operator to select a symbolic file member when a date or time mismatch is detected for a COBOL or PL/I module being NEW copied or when an Assembler module being new copied has more that one member in the symbolic files.

Note: This parameter should be used with *caution* as it forces CA InterTest for CICS to use the first symbolic file member it finds for a program being NEW copied when breakpoints are reset after the new module is copied.

Set System-Wide Options

To activate the automatic breakpoint facility for the entire system, enter:

```
CNTL=ABP,ON,termid
```

Breakpoint displays are routed to the terminal where the program is running. Specify a terminal ID to indicate where breakpoint displays should be routed for non-terminal attached tasks and for tasks executing from non-3270 terminals.

Note: The delivered default is to activate the global automatic breakpoint facility when CA InterTest for CICS is initialized.

To deactivate the global automatic breakpoint facility, enter:

```
CNTL=ABP,OFF
```

If you deactivate the global automatic breakpoint facility, set the local Automatic Breakpoint option for one or more programs, transactions, or terminals. The local Automatic Breakpoint option overrides the system automatic breakpoint facility.

To log all CNTL commands (except INQ and LIST) and all changes made to main storage, enter:

```
CNTL=GLOG,ON
```

To de-activate global logging of CA InterTest for CICS commands, enter:

```
CNTL=GLOG,OFF
```

To activate checkpointing so that the status of the CA InterTest for CICS monitoring is periodically monitored so that you can use the restart feature, enter:

```
CNTL=CKPT,INTRVAL=hhmm
```

The first checkpoint occurs when the option is specified. Subsequent checkpoints occur at the end of the interval, specified in hours and minutes. The time interval must be at least ten minutes,

To terminate checkpointing, enter:

```
CNTL=CKPT,OFF
```

To periodically purge all tasks waiting at a breakpoint, enter:

```
CNTL=PURGE,INTRVAL=hhmm
```

The first purge occurs when the option is specified. Subsequent purges occur at the end of the interval, specified in hours and minutes. The time interval must be at least 20 minutes

To purge all tasks waiting at a breakpoint at once, enter:

```
CNTL=PURGE,ALL=ONCE
```

To deactivate the purge facility, enter:

```
CNTL=PURGE,OFF
```

Example

To activate checkpointing with the first checkpoint occurring immediately and subsequent checkpoints occurring at 50 minute intervals, enter:

```
CNTL=CKPT,INTRVAL=0050
```

Execute a Module of CNTL Commands

Assemble modules containing CNTL commands by using the CA InterTest for CICS PROMMAC macros.

Note: For details, see the *Installation Guide*.

After the product is activated, enter the following command to execute the commands in the module:

```
CNTL=EXEC,MODULE=modulename
```

Note: The CNTL=LIST and CNTL=INQ commands cannot be specified in the module.

Construct as many of these modules as you want. Each should have its own entry in the PPT.

CNTL Monitoring Options

The following table describes the CNTL monitoring options and indicates the function to select from the Function Selection Menu to specify the same option.

Option	Description	CNTL Menu Function
ABI	Intercepts all CICS abends.	Not available
ABP	Sets local automatic breakpoint.	Function 22
BYP	Bypasses storage protection, permit SVC, BALR 14,14 and BALR 14,15 instructions.	Function 21
CBP	Sets or removes conditional breakpoints.	Function 12
CMD	Sets or removes indirect command statements for a program/terminal.	Function 24
CSA	Unprotects area in CSA.	Function 21
CWA	Unprotects area in CWA.	Not available
DM	Sets or removes data monitoring	Function 16
FEP	Identifies instructions that read certain main storage areas.	Not available
FOL	Continues monitoring after branch to another program.	Function 22
ICT	Instruction counter for preventing AICA abends.	Not available

Option	Description	CNTL Menu Function
KEP	Keeps data area in a source listing Keep window.	Not available
LET	Allows modification of storage or load module.	Function 21
LNK	Sets composite support.	Function 23
MON*	Sets or removes monitoring on locations for segmented monitoring.	Function 14
MUS	Limits number of times a program is monitored.	Function 22
MXR	Limits number of CICS requests.	Function 22
MXS	Limits storage usage.	Function 22
NOM*	Sets or removes monitor off locations for segmented monitoring.	Function 15
NRB	Prevents a read buffer before a breakpoint display.	Not available
NUP	Prevents program from updating files.	Function 22
OVR	Overrides errors that trigger an automatic breakpoint.	Not available
PRO	Protects storage from being modified.	Function 21
RBP	Sets or removes request breakpoints.	Function 13
RFC	Replaces file name.	Function 20
RNT	Prevents program from modifying code.	Function 22
RPC	Replaces program name.	Function 20
RTD	Replaces transient data queue name.	Function 20
RTS	Replaces temporary storage ID.	Function 20
SDF	Structure Display Format Setting	Function 22
SLB	Activates Source Listing Breakpoint display.	Function 22
STR	Saves CICS trace table.	Not available
TAL	Counts how often instruction is executed within loop.	Not available
TER	Changes the terminal ID.	Not available
TON	Limits monitoring to single terminal.	Functions 20,21,22
TRC	Sets or removes statement tracing	Function 11

Option	Description	CNTL Menu Function
UBP	Sets or removes unconditional breakpoints.	Function 11
USH	Removes storage protection from SHARED storage subpool area.	Not available
USR**	Sets monitoring for the CICS user ID specified.	All monitoring and most monitoring options

* Release 5.1 and higher only.

** Release 5.4 and higher only.

CNTL Monitoring Option Syntax

To set CNTL options, enter:

CNTL=ON,promid,options or **CNTL=ON*,options**

To remove CNTL options, enter:

CNTL=OFF,promid,options or **CNTL=OFF*,options**

Note: To remove an option, specify it exactly as it was originally defined.

To remove an option that was specified more than once for a program, enter:

CNTL=OFF,promid,option=.ALL

The .ALL specification is valid for the following options: BYP, CBP, CSA, KEP, LET, MON, NOM, PRO, RFC, RPC, RTD, RTS, and UBP.

Examples

To instruct CA InterTest for CICS to limit the number of CICS requests made by program COBDEMO to 40, enter:

CNTL=ON,PROG=cobdemo,MXR=40

To instruct CA InterTest for CICS to stop limiting the number of CICS requests made by program COBDEMO for user ID BARNEY1, enter:

CNTL=OFF,PROG=cobdemo,MXR=40,USR=BARNEY1

To instruct CA InterTest for CICS to remove all RFC options specified for program COBDEMO, enter:

CNTL=OFF,PROG=cobdemo,RFC=.ALL

Multiple Specifications of the Same Option

Multiple specifications of the same option are in effect for a program, transaction, or terminal.

Example

Specify the UBP option at various times to set unconditional breakpoints within a program. Each specification remains in effect until it is removed.

Specify the following options more than once: BYP, CBP, CSA, DM, KEP, LET, MON, NOM, PRO, RFC, RPC, RTD, RTS, SLB, TRC, and UBP. For other options, only one specification is in effect at one time for a program, transaction, or terminal. The last specification is the one that CA InterTest for CICS recognizes.

For the following options, only one specification is in effect at a time: ABI, ABP, FOL, ICT, MUS, MXR, MXS, NUP, RNT, STR, TAL, and TON.

ABI: Intercept All CICS Abends

The ABI option intercepts CICS abends and produces automatic breakpoints with an error code and message explaining the reason for the abend. Then select to continue execution (after, possibly, removing the error) or abend the task with the same or different abend code. All abend codes are intercepted, except those generated into the IN25ABEN table by you.

There are three ABI option values: ON OFF and FORCE.

ABI=ON (Default). Intercepts all CICS abends and produces automatic breakpoints for programs or tasks that *do not* have an active CICS HANDLE ABEND. If you want to intercept CICS abends for programs or tasks that have an active CICS HANDLE ABEND command, use the ABI=FORCE option.

ABI=OFF causes CA InterTest for CICS to STOP intercepting CICS ABENDS for the specified program.

Examples

```
CNTL=ON,PROG=ABC,ABI=ON

CNTL=ON,PROG=ABC,ABI=OFF

CNTL-ON,PROG=ABC,ABI=FORCE
```

ABP: Local Automatic Breakpoint Option

The *local* ABP option activates automatic breakpoints for a specified program, transaction, or terminal.

To set the local automatic breakpoint option, enter:

ABP=termid or **ABP=***

termid

Specifies the ID of the terminal to which automatic breakpoint displays should be routed.

*****

Specifies that the current terminal should receive automatic breakpoint displays.

This option overrides the *global* automatic breakpoint facility and is useful in a system where the global automatic breakpoint facility is disabled.

The local ABP option is also useful when you do not want the breakpoint display routed to the terminal where the program is running, or when the display cannot be routed to that terminal (for example, a non-terminal task or a task running on a non-3270 terminal). In these cases, specify the *termid* of the terminal to which breakpoint displays should be routed.

Remove the ABP option to deactivate automatic breakpoints for a specified program, transaction, or terminal. If a monitoring violation occurs, CA InterTest for CICS abends the task instead of halting it at an automatic breakpoint.

BYP: Bypass Storage Protection

To bypass storage protection and allow SVC and BALR, BASSM, BASR 14,15 and 14,14 instructions, enter:

BYP=(offset1,offset2)

or

BYP=(addr1,addr2)

Specify the beginning and ending program locations either as offsets or addresses.

offset

Specifies a one to five digit hexadecimal displacement from the beginning of the program.

addr

Specifies a six- to eight-digit hexadecimal address.

This BYP specification defines a section of program code that is allowed to:

- Modify any area of storage
- Issue SVC instructions
- Issue BALR, BASSM, and BASR 14,15 and 14,14 instructions

WARNING! Use this option with great care! Specify BYP only for a small section of code, such as a CALL statement (that is, a BALR, BASSM, or BASR 14,15 or 14,14 instruction). In that case, monitoring is suspended until control returns to the instruction after the CALL statement. If control does not return, monitoring remains off for the duration of the program's execution.

This BYP specification suspends monitoring when a BALR, BASSM, or BASR 14,15 or 14,14 instruction passes control to the specified location. It is useful when a monitored program calls a non-standard interface from several different locations. Use the following specification to bypass storage protection when a BALR, BASSM, or BASR 14,15 or 14,14 instruction passes control:

BYP=(.ANY,locn)

locn

Specifies a one- to five-digit hexadecimal displacement or a six- to eightdigit hexadecimal address.

Use the following specification to bypass a current instruction at a breakpoint:

BYP=*

This BYP specification instructs CA InterTest for CICS to allow the current instruction to modify storage or issue an SVC, BALR, BASSM, or BASR 14,15 or 14,14 instruction.

Note: Specify this option with great care. It could be password protected.

CBP: Specify Conditional Breakpoints

To specify a conditional breakpoint, enter:

CBP=locn,IF=condition

A conditional breakpoint halts a program at a specified location only if a specified condition is met. For instructions on how to specify the location and the condition, Conditional Breakpoints (see page 346) later in this chapter.

CMD: Set Indirect Command Statements

To set an indirect command statement for a particular program or terminal, enter:

```
CMD=((number),F=termid),TEXT='command text'
```

number

Specifies the indirect command statement line number that will be assigned to the accompanying command text.

F=termid

Specifies the terminal where the program for which the indirect commands are being set will execute from. This provides multiple users with the potential for defining independent sets of indirect commands for each of their debugging sessions. In addition, a single set of commands is shared by all users if F=.ANY is specified.

Note: Indirect commands defined under a specific or .NO terminal take precedence over any existing F=.ANY command set.

TEXT

Specifies the actual indirect command text, which must be enclosed within single or double quotation marks. For details and command syntax, Indirect Commands (see page 366) later in this chapter.

CSA: Unprotect an Area in the CSA

To unprotect an area in the CSA, enter:

```
CSA=(offset,length)
```

offset

Specifies a hexadecimal displacement from the beginning of the CSA.

length

Specifies the number of hexadecimal bytes in the CSA to be unprotected.

This option lets a program, transaction, or terminal modify areas in the CSA.

Important! Specify this option with great care. It could be password protected.

CWA: Unprotect an Area in the CWA

This option lets a program, transaction, or terminal modify areas in the CWA.

To unprotect an area in the CWA, enter:

```
CWA=(offset,length)
```

offset

Specifies a hexadecimal displacement from the beginning of the CWA.

length

Specifies the number of hexadecimal bytes in the CWA to be unprotected.

DM: Data Monitoring Option

This option instructs CA InterTest for CICS to capture data values for every COBOL statement executed in the program or transaction, or at a terminal. When data monitoring is set to ON, the statement tracing option (TRC) is forced on for the program, transaction, or terminal.

To set data monitoring for a particular program, transaction, or terminal, enter:

```
DM=ON
```

Note: A specific terminal ID or user ID is required to set the data monitoring option. If one is not specified, the default will be used.

FEP: Set the FEP Option

The FEP option lets you find program instructions that read from one or more main storage areas specified in the option. The first format of the FEP option identifies main storage areas that a program might read. The second format (FEP=EX,...) prevents the program from reading *all* main storage areas *except* the ones specified in the option and areas owned by the program.

To set the FEP option, enter:

```
FEP=(addr,len)
FEP=(EX,addr,len)
```

addr

Indicates six to eight digit hexadecimal address of the first byte of a main storage area.

len

Indicates length of the area in hexadecimal.

EX

Prevents the program from reading all main storage areas except those specified in the option or owned by the program.

FOL: Continue Monitoring After a Branch to Another Program

This option instructs CA InterTest for CICS to monitor a program even after it branches directly to another program (wild branch). All monitoring options remain in effect.

Specify breakpoints in the load module to which control is passed, but their locations might have to be specified as hexadecimal addresses.

Note: Do not specify this option if control is passed using a LINK or XCTL instruction.

- Use FOL=ON when control passes to a program that does not have a PPT entry.
- Use FOL=name when control passes to a program that has a PPT entry.
- Use FOL=NOPPT instead of ON to reduce overhead.

To continue monitoring after a branch, enter:

```
FOL=ON
```

or

```
FOL=name
```

or

```
FOL=NOPPT
```

ICT: Instruction Counter for Preventing AICA Abends

This command instructs CA InterTest for CICS to issue a DFHKC TYPE=WAIT,DCI=CICS macro each time the specified program executes *nnnnn* instructions. Use this option to prevent incorrect AICA abends in high CPU-utilization programs where overhead was increased by the CA InterTest monitoring activity.

To set the instruction counter to prevent an AICA abend, enter:

ICT=nnnnn

nnnnn

Specifies the number of instructions. Replace *nnnnn* with a decimal number between 1 and 65,534.

KEP: Keep Data Areas in a Keep Window

The KEP= option lets you specify one storage area to be kept in a Keep window whenever the specified program displays or executes from the source listing facility. The Keep window is not available on a Detailed Breakpoint display.

Display up to six items in a Keep window at a time. If you keep more than six items, CA InterTest for CICS initially displays the first six items that you selected but allows scrolling to other items using the PF19 and PF20 keys. When an item is removed from the window, another item is displayed. The KEP= option syntax allows only one request at a time. For each item kept, CA InterTest for CICS displays a single line identifying the keep request, followed by the first 12 bytes of storage in both hexadecimal and character format.

To keep data in a Keep window, enter:

KEP=(request)

KEP=((request),T=termid)

request

Specifies one data area in CICS main storage to be displayed in a keep window. Use CORE syntax to specify the requested data area, but omit CORE=.

termid

Specifies the terminal that displays the requested area in a Keep window.

T=.ANY

Specifies the data area that will display in a Keep window on any terminal that displays or executes the program using LIST.

For more information:

Source Listing Facility (see page 47)
Accessing Main Storage: CORE (see page 375)

LET: Allow a Program to Modify Storage or a Load Module

To allow a program to modify storage, enter:

LET=(address,length)

address

Specifies the starting address of the area of storage that is modified as six to eight hexadecimal digits.

length

Specifies, in hexadecimal, the number of bytes that are modified.

The storage area must lie entirely within the CICS Dynamic Storage Area.

To allow a program to modify a load module, enter:

LET=progname

progname

Specifies the load module that is modified.

Note: Specify this option with great care. It could be password protected.

LNK: Set Composite Support

Use LNK options to set composite support for a *composite module*. A composite module consists of separately compiled or assembled parts brought together when the module is link-edited.

The part of the composite module that receives control from CICS is referred to as the *main* program; the remaining parts are referred to as *subprograms*. These programs are written in the same or different languages.

Composite support lets you test and debug a subprogram as if it were a separate program. Specify a separate LNK option for the main program and for each subprogram you want to test separately.

It is easier to set LNK options on the CNTL Composite Support screen than with single-line commands. Most of the necessary information is supplied in a CA InterTest for CICS batch job step.

To set composite support for a composite module, enter:

LNK=(composite module information)

For more information:

Monitoring Menu Options (see page 227)

MON: Set Monitor ON Locations for Segmented Monitoring

Important! The segmented monitoring options MON and NOM may cause unpredictable damage to the CICS system if not used properly. Instructions for using MON and NOM are in the chapter "Special Monitoring Situations." MON and NOM pertain to all users and are not user ID sensitive.

Notes:

MON is supported only in releases 5.1 and higher.

MON options are not user ID sensitive and pertain to all users.

To set monitor on locations for segmented monitoring, enter:

```
MON=locn
MON=(locn1,locn2,...,locn9)
```

locn

Specifies a **program** location where CA InterTest for CICS starts or continues monitoring using the special segmented monitoring method. The location must specify the op-code of a machine instruction that resides in non-store-protected virtual storage.

Specify, in quotes, the location as a COBOL paragraph name or a PL/I or Assembler label; without quotes a COBOL or PL/I statement number (a decimal number preceded by a # sign), or a hexadecimal displacement from the beginning of the program, or an address in main storage.

MUS: Limit the Number of Times a Program Is Monitored

To limit the number of times a program is monitored, enter:

```
MUS=limit
```

limit

Specifies the number of times a program is monitored. Replace *limit* with a decimal number from 1 to 65,534.

MXR: Limit the Number of CICS Requests

The MXR option is useful when a program is in a loop that includes a CICS request and therefore will not time out.

To limit the number of CICS request, enter:

MXR=nnnnnnn

nnnnnnn

Specifies the number of CICS requests a program issues. Replace *nnnnnnn* with a decimal number up to one million.

MXS: Limit Storage Usage

To limit storage usage, enter:

MXS=nnnnnnn

nnnnnnn

Specifies the amount of storage, including program storage that a program acquires. Replace *nnnnnnn* with a decimal number up to one million.

NOM: Set NO Monitor Locations for Segmented Monitoring

Important! The segmented monitoring options MON and NOM might cause unpredictable damage to the CICS system if not used properly. Instructions for using MON and NOM are in the chapter "Special Monitoring Situations." MON and NOM pertain to all users and are not user ID -sensitive.

Notes:

NOM is supported in releases 5.1 and higher, only

NOM is not user ID sensitive and pertains to all users

To set no monitor locations for segmented monitoring, enter:

NOM=locn
NOM=(locn1,locn2,...,locn9)

locn

Specifies a *program* location where CA InterTest for CICS stops monitoring using the special segmented monitoring method. The location must specify the op-code of a machine instruction that resides in non-store-protected virtual storage.

Specify the location as a COBOL paragraph name (in quotes), a PL/I or Assembler label (in quotes), a COBOL or PL/I statement number (a decimal number preceded by a # sign), or a hexadecimal displacement from the beginning of the program. The location also specifies an address in main storage.

NRB: Prevent a Read Buffer Before a Breakpoint Display

The read buffer request lets CA InterTest for CICS restore the screen when you are finished with the breakpoint display. However, if the read buffer cannot be executed because the terminal or transmission network prevents it, you must issue the NRB option to prevent the request.

To prevent CA InterTest from issuing a read buffer before displaying a breakpoint, enter:

```
NRB=ON
```

NUP: Prevent a Program from Updating CICS Files

The NUP option prevents a program from actually updating VSAM and BDAM CICS files. This option allows repeated tests of transactions without altering test records. The NUP option also prevents test programs from modifying production files.

To prevent a program from updating CICS files, enter:

```
NUP=ON
```

OVR: Override Error Conditions That Trigger an ABP

The OVR option instructs CA InterTest for CICS to ignore error conditions that would otherwise trigger an automatic breakpoint. For a list of error codes, see the appendix "Examining Dumps with CA InterTest for CICS."

To override an error that could trigger an ABP, enter:

```
OVR=errcode
```

or

```
OVR=(errcode1,errcode2,...)
```

errcode

Specifies a hexadecimal CA InterTest for CICS error code.

Important! Use caution in specifying this option. Remember, it forces CA InterTest for CICS to ignore potentially dangerous conditions.

For more information:

Examining Dumps (see page 559)

PRO: Protect Storage from Being Modified

The PRO option generates an automatic breakpoint whenever the monitored program attempts to modify the protected area. This is useful when you are trying to determine how an area of storage is being modified.

Specify the PRO option while a program is at a breakpoint if the storage location is different each time the program executes.

To protect storage from being modified, enter:

PRO=(address,length)

address

Specifies the starting address of the protected storage area as six- to eight-hexadecimal digits.

length

Specifies, in hexadecimal, the number of protected bytes.

RBP: Specify Request Breakpoints

The RBP option lets you halt a program prior to all CICS commands, macros, DL/I calls, or calls to software recognized by CA InterTest for CICS. This option also lets you halt a program prior to specific CICS commands, such as File Control or Program Control commands. Most users will find it easier to use the CNTL Command Builder menus rather than the RBP option to specify request breakpoints.

Note: If the F=term1 or T=term2 parameter is omitted, it defaults to the terminal that issued the RBP option.

To specify request breakpoints, use the following commands:

```
RBP=request
RBP=(request1,request2,...,request9)

RBP=((request),F=term1,T=term2)
RBP=((request1,...,request9),F=term1,T=term2)

RBP=((request),F=term1,T=term2,L=loopnumber)
RBP=((request1,...,request9),F=term1,T=term2,L=loopnumber)
```

request

Specifies where the request breakpoint should occur:

- ALLCOM prior to all CICS commands
- ALLMAC prior to all CICS macros
- ALLDLI prior to all DL/I calls
- ALLCAL prior to all calls to software CA InterTest for CICS has been instructed to recognize at installation time
- *code* prior to the CICS commands or macros represented by the code

For a list of valid codes, see the Help facility (see page 460).

term1

Specifies the terminal at which the program must be executing for the breakpoint to take effect. F=.ANY specifies that the breakpoint take effect at all terminals—even when the program executes without a terminal. F=.NO specifies that the breakpoint take effect only when the program executes without a terminal.

term2

Specifies the terminal to which the breakpoint display is sent. T=.ANY specifies that the breakpoint display be sent to the terminal at which the program is executing.

loopnumber

Specifies how often the program should be halted at the breakpoint.

Example

L=2 halts the program every other time it reaches the specified command; L=13 halts it every thirteenth time. If this parameter is omitted, the breakpoint occurs every time the program reaches the specified command or macro. This parameter is especially useful when a CICS command occurs within a loop; for example, during a browse.

RFC: Replace File Names

The RFC option dynamically substitutes one file name for another during program execution. This is useful when you want to use test files in a production environment, or when you want to use several different files to test a program.

To replace file names, enter:

RFC=(file1,file2)

file1

Specifies the file to be replaced.

file2

Specifies the file that will replace *file1*.

RNT: Prevent a Program from Modifying Its Own Code

The RNT option lets you test a program for reentrancy by preventing the program from modifying its own code.

To prevent a program from modifying its own code, enter:

RNT=ON

RPC: Replace Program Names

The RPC option dynamically substitutes one program name for another in LOAD, LINK, and XCTL requests made by a monitored program. This lets multiple copies of a program coexist in one system, provided each has its own PPT entry. Use the RPC option when you want to test a program using modules other than the ones coded in the program.

To replace program names, enter:

RPC=(prog1,prog2)

prog1

Specifies the program to be replaced.

prog2

Specifies the program to replace *prog1*.

RTD: Replace Transient Data Queue Names

The RTD option dynamically substitutes one transient data queue name for another during program execution so you can test a program using transient data queues other than the ones coded in your program.

To replace transient data queue names, enter:

RTD=(name1,name2)

name1

Specifies the transient data queue name to be replaced.

name2

Specifies the transient data queue name to replace *name1*.

RTS: Replace Temporary Storage Identifications

The RTS option dynamically substitutes one temporary storage identification for another during program execution so you can test a program using temporary storage identifications other than the ones coded in your program.

To replace temporary storage IDs, enter:

RTS=(mask1,mask2)

mask1

Specifies the temporary storage ID to be replaced.

mask2

Specifies the temporary storage ID to replace *mask1*.

Specify both masks as eight bytes in either character (C' data') or hexadecimal (X' data') format.

SDF: Structure Display Format Setting

To override, by program, the global default setting, enter:

SDF=HEX or SDF=DATA

HEX

Displays data in hexadecimal or character format.

DATA

Displays data in Structure Display Format.

SLB: Activate the Source Listing Breakpoint Facility

This option activates the Source Listing Breakpoint display for the specified terminal.

To activate the source listing breakpoint facility, enter:

SLB=termid
SLB=*

termid

Specifies the terminal to receive breakpoint displays.

*****

Specifies the terminal from which the option is issued.

STR: Save the CICS Trace Table

The STR option instructs CA InterTest for CICS to preserve a copy of the CICS trace table when a program stops at a breakpoint. Issue the command CORE=STRA to display the table. The most recent table entries display first; view earlier entries by scrolling through the display.

To save the CICS trace table, enter:

STR=ON

TAL: Count How Often an Instruction Is Executed

The TAL option counts the number of times a program reaches a specified location. This option is useful with a conditional breakpoint because it lets you specify how often a program should be halted within a loop. Use the *loopnumber* parameter when specifying an unconditional breakpoint to perform the same task.

To count the number of times an instruction executes, enter:

TAL=locn

locn

Specifies a *program* location as a COBOL paragraph name (in quotes), a PL/I or Assembler label (in quotes), a COBOL or PL/I statement number (preceded by a # sign), or a hexadecimal displacement from the beginning of the program.

TER: Change the Terminal ID

The TER option changes all occurrences of the old terminal ID to the new terminal ID whenever a terminal ID is specified for an online option; for example, in a UBP option. This is useful when you have set terminal-specific options, and then need to move to another terminal to continue testing.

To change the terminal ID, enter:

TER=(oldtrm,newtrm)

oldtrm

Specifies the old terminal ID.

newtrm

Specifies the new terminal ID.

TON: Limit Monitoring to One Terminal

The TON option instructs CA InterTest for CICS to monitor a program or transaction only when it executes at a specific terminal or without a terminal.

To limit monitoring to a single terminal, enter:

TON=termid

or

TON=*

or

TON=.NO

termid

Specifies the terminal the program must execute from to be monitored.

*****

Specifies that the program must execute at the current terminal to be monitored.

.NO

Specifies that the program must execute without a terminal to be monitored.

TRC: Statement Tracing Option

This option instructs CA InterTest for CICS to save trace information for every COBOL statement executed in the program or transaction, or at a terminal.

To set statement tracing for a particular program, transaction, or terminal, enter:

```
TRC=ON
```

This option is unrelated to the backtrace facility.

UBP: Specify Unconditional Breakpoints

To specify unconditional breakpoints, use the following commands:

```
UBP=locn
UBP=(locn1,locn2,...,locn9)

UBP=((locn),F=term1,T=term2)
UBP=((locn1,...,locn9),F=term1,T=term2)

UBP=((locn),F=term1,T=term2,L=loopnumber)
UBP=((locn1,...,locn9),F=term1,T=term2,L=loopnumber)

ubp=((ALLPAR),F=term1,T=term2,L=loopnumber)
ubp=((ALLLAB),F=term1,T=term2,L=loopnumber)
```

locn

Specifies a program location as a COBOL paragraph name (in quotes), a PL/I or Assembler label (in quotes), a COBOL or PL/I statement number (preceded by a # sign), or a hexadecimal displacement from the beginning of the program. locn also specifies an address in main storage.

The following operands specify the terminal at which the program must be executing for the breakpoint to take effect.

- F=.ANY specifies that the breakpoint take effect at all terminals, even when the program executes without a terminal.
- F=.NO specifies that the breakpoint take effect only when the program executes without a terminal.

term2

Specifies the terminal to which the breakpoint display is sent.

T=.ANY

Specifies that the breakpoint display be sent to the terminal at which the program is executing.

Note: If the F=term1 or T=term2 parameter is omitted, it defaults to the terminal that issued the UBP option.

loopnumber

Specifies how often the program should halt when a breakpoint is within a loop.

Example

L=2 halts the program every other time it passes through the loop; L=3 halts it every third time. If this parameter is omitted, the breakpoint occurs every time the program reaches the location.

ALLPAR

Specifies that breakpoints take effect at *all* COBOL paragraph names.

ALLLAB

Specifies that breakpoints take effect at *all* Assembler labels.

When you set unconditional breakpoints, CA InterTest for CICS halts the program at the specified locations.

USH: Unprotect Shared Storage

The USH option removes storage protection from all main storage areas residing in the SHARED storage subpool. This option lets a program change all CLASS=SHARED storage, and other storage areas in the SHARED subpool. FREEMAINs of these areas are permitted; however, the four-byte Storage Accounting Areas remain protected.

To unprotect shared storage, enter:

```
USH=ON
```

The TER option changes all occurrences of the old terminal ID to the new terminal ID whenever a terminal ID is specified for an online option; for example, in a UBP option. This is useful when you have set terminal-specific options, and then need to move to another terminal to continue testing.

Important! Use this option with extreme caution.

USR: Limit Monitoring to a CICS User ID

When security is active in the region and the IN25OPTS global installation option is set to DFLTUSR=SPECIFIC, a signed on user's monitoring request defaults to using the requester's individual user ID (USR=userid). Otherwise, the default USR= option is set to .ANY.

Monitoring options set for a specific user ID have advantages. The From and To terminals automatically default to .ANY. This means that you can switch terminals and continue to have monitoring, breakpoints, and options set for his user ID directed to the new terminal.

For releases 5.4 and higher, to limit monitoring to a specific CICS user ID in a secure region, enter:

USR=userid

userid

Specifies CICS user ID.

To monitor all users of the *promid*, enter:

USR=.ANY

CNTL Capabilities

Use the CNTL transaction to perform these functions:

- Monitoring
- Breakpoints

Monitoring

Set and remove monitoring for one or more programs, all the programs executed by a transaction, or all the programs executed from a terminal.

Breakpoints

Set and remove unconditional or conditional breakpoints.

Set and remove request breakpoints prior to CICS macros and commands, calls to DL/I or DB/2, and calls to database software.

Monitoring Options

With the CA InterTest for CICS CNTL feature, you can:

- Set and remove statement tracing options for COBOL programs, including data monitoring
- Dynamically replace one or more CICS resources (programs, files, temporary storage, transient data queues) used by a program

- Override default protection rules
- Set and remove special monitoring options, such as limiting the number of times a program is monitored or preventing a program from updating a file
- Provide symbolic support for programs that consist of separately compiled modules brought together when the program is link-edited (composite support)

Status Display, Utility, and System Options

With the CA InterTest for CICS CNTL feature, you can:

- Display the status of monitored programs, transactions, or terminals
- Display the contents of CA InterTest for CICS tables and files
- Reset monitoring for a recompiled program
- Establish monitoring options for the entire system

Resume Task Execution

With the CA InterTest for CICS CNTL feature, you can resume execution of a task halted at a breakpoint.

Note: Use the CNTL menus and screens or CNTL commands to perform all of the functions listed in the previous two sections. Additional options, specified by commands, are available to perform more specialized functions.

Initiate Monitoring

CA InterTest for CICS provides many ways of initiating monitoring and performing other CNTL functions:

- Source listing facility
- ITST menus and fill-in screens
- CNTL menus and fill-in screens
- CNTL commands

With the CNTL facility use one method exclusively or use the mix-and-match method—even during one test session, depending on your specific testing needs.

Example

Use the source listing facility to initiate monitoring and set breakpoints, the CNTL menus and screens to set special options, and a single-line command to resume execution of a task.

The easiest way to turn on monitoring and set breakpoints for a single program is to use the source listing facility. For a new or infrequent user, this method is strongly recommended. Turn on monitoring by pressing a PF key, and set a breakpoint by entering a single keystroke directly on your program's source listing. The chapter "Source Listing Facility" discusses this facility and gives detailed information on what to do when your program halts at a breakpoint.

To take advantage of the full range of CA InterTest for CICS monitoring capabilities, use the CNTL Monitoring Command Builder menus and screens or enter CNTL commands. Nearly all the monitoring functions are implemented either way, so select the method that suits you best.

The CNTL Monitoring Command Builder menus and screens make it easy for you to specify the necessary information without worrying about command syntax. All you have to do is select the function you want and enter the data—CA InterTest for CICS creates the right command for you. Online Help is available from every screen using PF1 to provide you with more information.

If you prefer, enter single-line CNTL commands to specify monitoring functions. More experienced users often find it faster to enter commands directly.

To access online Help when entering single-line commands, type Help on a clear screen. CA InterTest for CICS displays the Transaction Codes screen. Press **Enter** and CA InterTest for CICS displays the help facility master menu. Two main topics (program monitoring and breakpoints) and their associated subtopics provide information on CNTL commands.

Monitor Command Building Menus and Screens

This section describes how to use the CNTL Monitoring Command Builder menus and screens to perform the following CNTL functions. The order in which CNTL functions are described corresponds to their order on the Function Selection Menu.

Function Selection Menu—The main menu for selecting CNTL functions.

Setting and Removing Monitoring—Begin and end monitoring for programs, transactions, or terminals.

Setting and Removing Unconditional Breakpoints—Set and remove unconditional breakpoints.

Setting and Removing Conditional Breakpoints—Set and remove conditional breakpoints.

Setting and Removing Request Breakpoints—Set and remove request breakpoints before CICS macros and commands, calls to DL/I, DB2, and SQL/DS, and calls to database software.

Setting and Removing Statement Trace Options—Set and remove statement tracing and data monitoring for COBOL programs.

Setting and Removing Replacement Options—Dynamically change CICS resources: programs, files, transient data queues, temporary storage.

Setting and Removing Protection Options—Bypass CA InterTest for CICS rules that protect main storage, the CSA, and load modules.

Setting and Removing Special Options—Modify monitoring rules (for example, prevent a program from updating a file, allow a program to branch to another program, limit the number of times a program is monitored).

Setting and Removing Composite Support—Provide symbolic support for a program that consists of separately compiled modules.

Status Display—Display the monitoring status of a program, transaction, or terminal.

Utility Options—Display the contents of CA InterTest for CICS tables and files, and reset monitoring for a recompiled program.

System-Wide Options—Request monitoring options on a system-wide basis.

Resume Task Execution—Resume execution of a task stopped at a breakpoint.

Access the Monitoring Command Builder Main Menu

To display the Monitoring Command Builder main menu (the Function Selection Menu), enter **CNTL** on a clear screen, or enter **CNTL** on the command line of any source listing, detailed breakpoint, or CORE screen.

For more information:

Monitoring Menu Options (see page 227)

The Function Selection Menu

The following Function Selection screen is the main menu for the Monitoring Command Builder. This menu lists the different CNTL functions detailed later in this chapter.

```
          CA InterTest MONITORING COMMAND BUILDER  FUNCTION SELECTION

 Select function by number: __ ❶
 For functions 1022, enter S to set or R to remove: _❷

 10  Monitoring                20  Replace Options    30  Status Display
 11  Unconditional Breakpoints 21  Protect Options    31  Utility Functions
 12  Conditional Breakpoints   22  Special Options    32  SystemWide Functions
 13  Request Breakpoints       23  Composite Support  33  Resume Task Execution
 14  Segmented Monitoring On   24  Indirect Commands
 15  Segmented Monitoring Off
 16  Statement Trace Options

 For functions 1030, enter Program Names or Transaction Codes or Terminal Names

 ❸ Program Names      COBDEMO  ________ ________ ________ ________ ________
    Transaction Codes  ____ ____ ____ ____ ____ ____ ____ ____ ____
    Terminal Names     ____ ____ ____ ____ ____ ____ ____ ____ ____

❹ user ID            ________  user ID (or .ANY) who will execute the program


 1 Help       2           3 End       4 Return    5              6
 7            8           9          10          11             12
```

To select a function, first enter the number of the desired function in field ❶.

If you select this function:	Also enter this:
10 through 22	In field ❷, enter S or R to specify SET or REMOVE. If left blank, it defaults to S.
10, 13, 16, 20, 21, 22, 24, or 30	In area ❸, enter the program names, transaction codes, or terminal names. Specify from one to six programs, one to nine transactions, or one to nine terminals, but you cannot combine different types of entries. Specify .ALL to indicate all programs, transactions, or terminals. Also specify .OPTIONS in the Program Names field to set options for all programs. Optionally, override the user ID default in area❹. Specify a CICS user ID, such as MARY01, or .ANY.
11, 12, 14, 15,or 23	In the Program Names field, enter one program name. Optionally, override the user ID default in area ❹. Specify a CICS user ID, such as MARY01, or .ANY.
31, 32, or 33	Do not enter anything in areas ❸ or❹.

Note: If a program is stopped at a breakpoint and you type **CNTL** in the command line to access the Function Selection Menu, CA InterTest for CICS displays the name of the monitored entry (program, transaction, or terminal) on this screen.

When you have entered all the necessary information, press **Enter**. In most cases, a second screen opens. If you have specified multiple programs, transactions, or terminals, the names of all the entries display on subsequent screens.

When you *set* breakpoints, replacement, protection, special options, or composite support, you automatically set monitoring for the programs, transactions or terminals that you specify. So, for example, it is not necessary to set monitoring and set breakpoints; setting breakpoints accomplishes both functions.

However, when you *remove* breakpoints, replacement, protection, special options, or composite support, you do not remove monitoring for the programs, transactions, or terminals that you specify. Monitoring remains in effect until you explicitly instruct CA InterTest for CICS to remove monitoring.

Generic Specification of Programs, Transactions, or Terminals

When you set monitoring or monitoring options for a specific program, transaction, or terminal, CA InterTest for CICS creates an entry for that program, transaction, or terminal in the monitoring table. Also set monitoring or monitoring options generically by specifying asterisks, *, and plus signs, +, to replace characters in a program name, transaction code, or terminal name. These specifications are interpreted according to the rules for the CICS CEMT transaction.

- An asterisk specifies any number of characters, including no characters. Specify more than one asterisk in a program name, transaction code, or terminal name. Do not, however, specify an asterisk by itself.
- A plus sign specifies just one character in a particular position. Specify more than one plus sign in a name or code. Do not, however, specify a plus sign by itself.
- Combine asterisks and plus signs.

Examples

Here is how the following specifications are processed:

ABC* Specifies all names or codes that begin with ABC, including ABC.

*ABC Specifies all names or codes that end with ABC, including ABC.

AB*C Specifies all names or codes that begin with AB and end with C with an unspecified number of characters in the middle.

ABC+ Specifies all four-character names or codes that begin with ABC followed by any one character.

+ABC Specifies all four-character names or codes that begin with any character followed by ABC.

AB+C Specifies all four-character names or codes that begin with AB followed by any one character and C.

AB+C* Specifies all names or codes that begin with AB followed by one character, C, and an unspecified number of characters.

Hierarchy Rules for Monitored Entries

It is important to know the priorities CA InterTest for CICS observes if more than one entry in the monitoring table applies to the same program.

- A specific or generic program entry overrides a transaction entry, a terminal entry, and an .ALL specification
- A transaction entry overrides a terminal entry and an .ALL specification
- A terminal entry (specific or generic) overrides an .ALL specification

Example

Suppose you specify monitoring options for program ABC1 and different options for transaction TRN1. ABC1 executes as part of that transaction. When ABC1 executes, the options specified for ABC1 override the options specified for TRN1. Similarly, it is important to understand the priorities CA InterTest for CICS observes when a program, transaction, or terminal is specifically and generically declared for monitoring. A specific entry overrides a generic entry *except* when you request Status Display. In this case, the generic specification takes precedence.

Suppose you specify monitoring options for program ABC1 and different monitoring options for program ABC*. The options specified for ABC* affect all programs whose names begin with ABC except for program ABC1 (and any other programs beginning with ABC that have their own entries in the monitoring table). Monitoring for ABC1 is controlled by the specific entry for that program. However, if you request a status display for program ABC*, you will get monitoring status reports for all programs that begin with ABC, including program ABC1 that has a separate monitoring table entry.

PF Keys

Exit from the Function Selection Menu by pressing a PF key. Most of these PF keys are also in effect on other Monitoring Command Builder screens. The following table describes the function of each PF key.

PF Key	Function
PF1 or 13	Initiates the CA InterTest for CICS online help facility.
PF2 or 14	Unassigned
PF3 or 15	Terminates the Function Selection Menu screen and returns to the active breakpoint, if appropriate. Pressing Clear does the same thing.
PF4 or 16	Returns to the source listing display facility.
PF5 or 17	Unassigned
PF6 or 18	Unassigned
PF7	Unassigned
PF8	Displays your previous CNTL Monitoring Command Builder screen with the specified data.
PF9 to end	Unassigned

For more information:

Accessing Main Storage: CORE (see page 375)
Accessing Auxiliary Storage: FILE (see page 441)

Monitoring

Set and remove monitoring for programs, transactions, or terminals directly from the Function Selection Menu. Specify function **10**, **S** or **R**, and the program names, transaction codes, or terminal names, and press Enter. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter. This is the only screen you need to complete.

Set Monitoring

When you set monitoring for a:

Program—CA InterTest for CICS detects and prevents errors *before* they occur. That means monitored programs will not abend or commit storage violations. CA InterTest for CICS monitors the program from whatever terminal it executes, even if it executes as part of a non-terminal owning task. Use the replacement, protection, or special options screen if you want to restrict monitoring to a specific terminal.

Transaction—CA InterTest for CICS monitors all of the transaction's programs.

Terminal—CA InterTest for CICS monitors every program that executes at that terminal.

Note: CA InterTest for CICS will not monitor programs that begin with the prefix DFH (IBM CICS programs) or IN25 (CA InterTest for CICS programs).

If you set monitoring for a transaction or terminal, and a program executing as part of that transaction or at that terminal is stopped at an automatic breakpoint, CA InterTest for CICS automatically creates an individual entry in the monitoring table for that program. This lets you set breakpoints and monitoring options specifically for that program. Similarly, if you specify .ALL in the Program Names field on the Function Selection Menu or if you specify a generic program name, CA InterTest for CICS creates an individual monitoring table entry for any program monitored under that entry when it stops at an automatic breakpoint.

Set the User ID for Monitoring

Monitoring is sensitive to CICS user IDs.

For more information:

Monitoring Menu Options (see page 227)

Remove Monitoring

Before you remove monitoring, you might want to produce a report of all CA InterTest for CICS monitoring activity to make sure other users are not currently testing the same programs. Press PF12 Status to produce a monitoring report.

When you remove monitoring, you also remove all breakpoints and monitoring options specified for that program, transaction, or terminal. A quick way of creating a new testing scenario for a program is to remove and then reset monitoring with different breakpoints and options.

Note: Remember that you can also set monitoring by setting breakpoints, replacement, protection, special options, or composite support. However, if you remove breakpoints, options or composite support, monitoring *remains in effect* for the specified programs until you specifically instruct CA InterTest for CICS to stop monitoring.

Unconditional Breakpoints

When you set unconditional breakpoints, CA InterTest for CICS halts the program at the specified locations. CA InterTest also automatically monitors that program.

To set or remove unconditional breakpoints, specify function **11, S** or **R**, and the program name on the Function Selection Menu, and press Enter. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

Set Unconditional Breakpoints

If you want to *set* unconditional breakpoints, the Breakpoint Locations screen opens. COBOL, PL/I, and Assembler versions of the Breakpoint Locations screen are available. The following screen is the COBOL version.

```
      CA InterTest MONITORING COMMAND BUILDER  COBOL BREAKPOINT LOCATIONS   11

   SET breakpoint locations for PROG=COBDEMO  in any of the following fields:
   (For qualification by nested program name, specify 'ProgramID:Label')
❶Para/Data _____________________________       _____________________________________
     Names: _____________________________        _____________________________________
            _____________________________        _____________________________________
  Statement
❷ Numbers: ____   _____   _____   _____   _____   _____   _____   _____

❸  Offsets: ____   _____   _____   _____   _____   _____   _____   _____

            All paragraph names:  _  ❹

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

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

To set breakpoints, enter the following information:

Specify up to nine breakpoint locations in fields ❶ through ❸.

COBOL

For a COBOL program, this is a paragraph name or data name ❶, a statement number ❷, or the hexadecimal displacement (offset) from the beginning of the

program ❸.

ASSEMBLER

For an Assembler program, breakpoint locations are Assembler labels, datanames, or offsets.

PL/I

For a PL/I program, breakpoint locations can be offsets or, if you have the PL/I symbolic option, PL/I labels or statement numbers.

Follow these rules when specifying information:

- If you specify statement #1, CA InterTest for CICS sets a breakpoint at the first executable instruction in a COBOL Procedure Division, Assembler CSECT, or PL/I procedure.
- COBOL statements can contain more than one verb. In specifying statement numbers, indicate the verb at which the breakpoint should occur. A statement number of nnn or nnn.0 indicates the first verb in the statement, nnn.1 indicates the second verb, nnn.2 indicates the third verb, and so on.

Example

A statement number entered as 503.3 specifies a breakpoint at the fourth verb in statement number 503.

- If you specify a COBOL or Assembler data name, breakpoints are set at every reference to that data name.
- Enter an X in the All paragraph names field (COBOL) ❹ or All Assembler labels field to set breakpoints at all paragraph names or all labels.
- Optional specifications:
- If you set a breakpoint within a loop, specify in field ❺ how often the program should be halted.

Example

If you specify 3, the breakpoint occurs every third time the program passes through the loop.

- Specify in field ❻ the ID of the terminal where the program must be executing for breakpoints to take effect. If you leave this field blank, it defaults to your current terminal. Enter .ANY to have breakpoints take effect at all terminals, even when the program executes without a terminal. Enter .NO to have breakpoints take effect only when the program executes without a terminal.
- Specify in field ❼ the ID of the terminal to receive breakpoint displays. If you leave this field blank, it defaults to your current terminal. Enter .ANY to have breakpoint screens displayed at the terminal where the program is executing when breakpoints occur.
- Specify in field ❽ the statement number of the indirect command that you want to invoke at the breakpoint you have set.
- Specify in field ❾ the user ID who must be executing the program for the breakpoint to take effect. Valid entries are .ANY or a specific CICS user ID.

When you complete entering all the necessary information, press **Enter**.

For more information:

Accessing Main Storage: CORE (see page 375)

Example

The following screen shows how to set unconditional breakpoints at four locations in the COBOL program COBDEMO.

```
CA InterTest MONITORING COMMAND BUILDER  COBOL BREAKPOINT LOCATIONS   11

    SET breakpoint locations for PROG=COBDEMO  in any of the following fields:
    (For qualification by nested program name, specify 'ProgramID:Label')
                          ↙
   Para/Data returntransid_________________      _________________________________

     Names: _________________________________      _________________________________

            _________________________________      _________________________________

  Statement   ↙         ↙         ↙
   Numbers: 1____    136__    174.2    _____    _____    _____    _____    _____


   Offsets: _____    _____    _____    _____    _____    _____    _____    _____

            All paragraph names:  _

          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                  MARY01___
```

The information in the previous screen causes breakpoints to occur at:

- Paragraph RETURN-TRANSID
- Statement 1 (the beginning of the Procedure Division)
- First verb in statement 136
- Third verb in statement 174

Because the user ID field specifies a single user and the terminal ID fields are left blank, the breakpoints take effect at any terminal under which User MARY01 is executing COBDEMO.

Remove Unconditional Breakpoints

If you indicated that you want to remove unconditional breakpoints, the Remove Breakpoint Locations screen opens. COBOL, PL/I, and Assembler versions of the Breakpoint Locations screen are available. The following screen is the COBOL version.

```
CA InterTest MONITORING COMMAND BUILDER   COBOL BREAKPOINT LOCATIONS  11

 REMOVE breakpoint locations for PROG=COBDEMO  in any of the following fields:
    (For qualification by nested program name, specify 'Program ID:Label')
❶Para/Data _________________________________    _________________________________
      Names: _________________________________    _________________________________
             _________________________________    _________________________________
  Statement
❷ Numbers: _____   _____   _____   _____   _____   _____   _____    _____

❸ Offsets: _____   _____   _____   _____   _____   _____   _____    _____

          All paragraph names:  _  ❹

          Term ID (or .ANY or .NO) where breakpoints will take effect: ____❺
          Term ID (or .ANY) that will receive the breakpoints:         ____❻

          user ID (or .ANY) who will execute the program               ______ ❼
 To remove by ID # shown in status display enter ID #'s: __ __ __ __ __ __ __ ❽
```

To remove a breakpoint, specify the information that you originally defined in fields ❶ through ❼ on the Set Breakpoint Locations screen, or enter its ID number in field ❽.

The ID number appears on the Detailed Breakpoint display. Press **PF12** Status to request a status report for the program. This report lists the ID of each breakpoint.

Note: To remove all unconditional breakpoints from a program, enter .ALL in field ❶, and press **Enter**.

Conditional Breakpoints

When you set conditional breakpoints, CA InterTest for CICS halts the program at the specified locations *only* if a condition is met. Conditional breakpoints are very useful in isolating complex problems.

To set or remove conditional breakpoints, specify function **12, S** or **R**, and the program name on the Function Selection Menu, and press Enter. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

Set Conditional Breakpoints

First, CA InterTest for CICS displays the Conditional Locations screen.

This screen is similar to the screen used to specify unconditional breakpoints, and you should enter the information in the same way. For details, see Set Unconditional Breakpoints (see page 108) earlier in this chapter.

A special feature lets you set a conditional breakpoint at *all* instructions.

To set conditional breakpoints:

1. Enter an x in the All Instructions field to request this option, as shown in the following example. If you request the All Instructions option and set a variable-change breakpoint, the breakpoint takes effect whenever the value of the variable changes.

```
CA InterTest MONITORING COMMAND BUILDER  COBOL CONDITIONAL LOCATIONS  12

   SET breakpoint locations for PROG=COBDEMO  in any of the following fields:

  (For qualification by nested program name, specify 'ProgramID:Label')
 Para/Data ________________________________     ________________________________

    Names: ________________________________     ________________________________

           ________________________________     ________________________________

 Statement
  Numbers: _____   _____   _____   _____   _____   _____   _____   _____

  Offsets: _____   _____   _____   _____   _____   _____   _____   _____

                                                                  ↙
          All paragraph names:  _            All instructions: x

        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. Specify the breakpoint locations, and press **Enter** to display the Conditional Breakpoint screen.
3. Specify the condition that must be satisfied for the breakpoint to take effect on the Conditional Breakpoint screen.

There are two versions of the Conditional Breakpoint screen: a simplified version for COBOL users, and a detailed version for PL/I and Assembler users and for those COBOL users who prefer this version.

On either version you, specify a condition in this format:

```
left side relational operator right side
```

If you are a COBOL user, the following screen opens.

```
CA InterTest MONITORING COMMAND BUILDER  CONDITIONAL BREAKPOINT

❶ Enter LEFT SIDE
       Data Name  taskswitch_____________________________________________


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

   Enter RIGHT SIDE
       Data Name _____________________________________________________________
          OR
❸    Literal 3_____________________________________________________________



            Press PF9 to go to complex conditional screen if necessary
```

- Field ❶specifies the left side of the comparison , which is a data name.
- Field ❷ specifies the relational operator:
  EQ for equal
  NE for not equal
  GT for greater than
  LT for less than
  GE for greater than or equal to
  LE for less than or equal to
- Field ❸specifies the right side of the comparison, which is a data name, literal, or figurative constant.

Variable-Change Breakpoints

- COBOL and Assembler users set a conditional breakpoint to take effect whenever the value of a variable changes. To do this, type **x** in the All Instructions field on the Conditional Locations screen.
- COBOL users must then specify the variable name in fields ❶ and ❸ on the Conditional Breakpoint screen.
- Assembler users must then specify the variable name in fields ❶ and ❺ on the Detailed Conditional Breakpoint screen, as discussed previously in The Detailed Conditional Breakpoint screen.

The operator entered in field ❷ determines whether the breakpoint occurs whenever the value of the variable changes (NE) or only when the value increases, decreases, or does not change. Whenever you re-execute the program, the initial value of the variable is reset to its value at the beginning of program execution.

Literal Formats

Specify four types of literals for the right side of the comparison:

Character	Specify character literals for comparisons with any field except a COBOL data name defined as COMP or COMP-3. Enter the literal as **C'character data'**, **X'hexadecimal data'**, or a combination of both. For example, C'ABC', X'0102', or C'ABC'X'0102'.
Packed	Specify packed decimal literals for comparisons only with COBOL data names defined as COMP-3. Enter the literal as **P'number'.** The number can be preceded by a minus sign. For example, P'123' or P'-123'.
Halfword	Specify halfword literals for comparisons with COBOL data names defined as COMP with a length of four, as in S9(4) COMP. Enter the literal as **H'number'**. The number can be preceded by a minus sign. For example, H'12' or H'-12'.
Fullword	Specify fullword literals for comparisons with COBOL data items defined as COMP with a length of eight, as in S9(8) COMP. Enter the literal as **F'number'**. The number can be preceded by a minus sign. For example, F'100' or F'-100'.

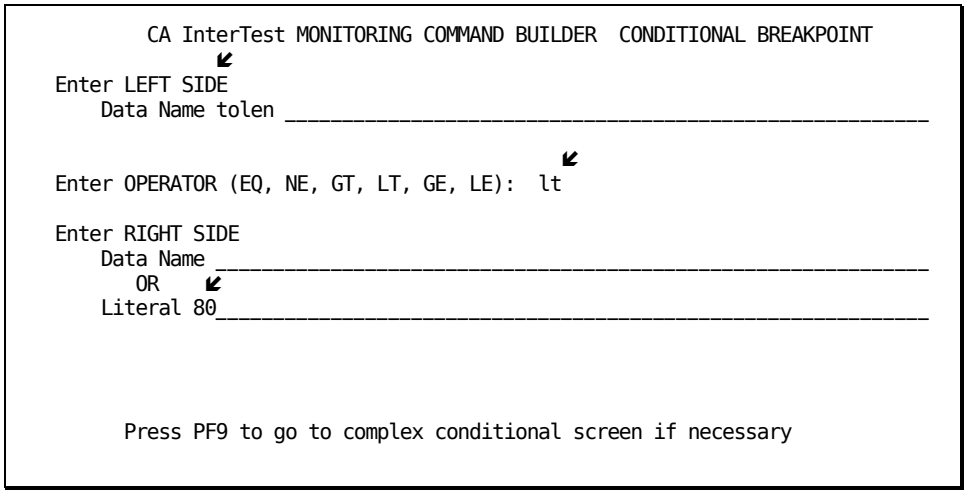
Figurative Constants

Specify the following figurative constants for the right side of the comparison:

Valid Right Side Specifications for the Comparison			
ZERO ZEROS ZEROES	numeric value 0	HIGH-VALUE HIGH-VALUES	character value X'FF'
SPACE SPACES	blank (X' 40')	LOW-VALUE LOW-VALUES	character value X'00'

Example

The following sample screen shows how to set a conditional breakpoint in a COBOL program.



- The left side of the comparison specifies the data name TOLEN.
- The relational operator is LT (less than).
- The right side of the comparison specifies the literal 80.

This breakpoint takes effect only if the value in the data name TOLEN is less than 80. The breakpoint locations are all the instructions, as previously specified on the Conditional Locations screen.

Remove Conditional Breakpoints

When you specify that you want to *remove* conditional breakpoints, the following Remove Conditional Locations screen opens.

```
CA InterTest MONITORING COMMAND BUILDER  COBOL BREAKPOINT LOCATIONS  11

 REMOVE breakpoint locations for PROG=COBDEMO  in any of the following fields:
    (For qualification by nested program name, specify 'ProgramID:Label')
❶Para/Data _____________________________    ______________________________
      Names: _____________________________    ______________________________
             _____________________________    ______________________________
  Statement
❷ Numbers: _____   _____   _____   _____   _____   _____   _____    _____

❸ Offsets: _____   _____   _____   _____   _____   _____   _____    _____

          All paragraph names:  _  ❹       All Instructions:___ ❺


          Term ID (or .ANY or .NO) where breakpoints will take effect: ____❻
          Term ID (or .ANY) that will receive the breakpoints:         ____❼

          user ID (or .ANY) who will execute the program               ______ ❽
```

The easiest way to *remove* a conditional breakpoint is from the Monitoring Status display, as discussed in the chapter "Menus and Displays." To remove *all* conditional breakpoints from a program, enter .ALL in field ❶of this screen, and press Enter.

The Detailed Conditional Breakpoint Screen

When Assembler and PL/I users want to specify conditional breakpoints complete the Conditional Locations screen and press **PF9**, the following screen opens. COBOL users access this screen by pressing **PF9** from the COBOL Conditional Breakpoint screen shown in the previous section.

```
          CA InterTest MONITORING COMMAND BUILDER  CONDITIONAL BREAKPOINT

❶  Enter LEFT SIDE of condition (select one):

     Data Name _____________________________________________________________
     Special keywords:   ______   Register # __         COBOL BLL # __


Enter OPERATOR (EQ, NE, GT, LT, GE, LE): __❷  Length:  Left  __❸ Right __❹

❺ Enter RIGHT SIDE of condition (select one):

     Data Name _____________________________________________________________
     Special keywords:   ______   Register # __         COBOL BLL # __
     Literal _______________________________________________________________

❻ Optional offset:  Enter +  @ OR % followed by a value.
     Left side  ____________________________   Right side ____________________

        Press PF9 to go to simple conditional screen
```

❶

Specifies the left side of the comparison must be an area of core. Enter information in *one* of these four fields:

- Data name
- Register
- COBOL BLL cell
- Area identified by a CORE keyword

❷

Specifies the relational operator:

- EQ for equal
- NE for not equal
- GT for greater than
- LT for less than
- GE for greater than or equal to
- LE for less than or equal to

❸/❹

Optionally define the length of the left side or right side of the comparison. For details, see Length of the Comparison (see page 353), which follows this section.

❺

Specifies the right side of the comparison, which is an area of core or a literal. Enter information in one of these five fields: a data name, a register, a COBOL BLL cell, an area identified by a CORE keyword, or a literal. For details, see CORE Keywords and Literal Formats (see page 118) later in this chapter.

❻

Optionally adjusts a CORE location by specifying offsets (displacements). Each offset must be preceded by one of the following operands:

- +
- -
- @ (indirect addressing below the 16-megabyte line) % (indirect addressing above the 16-megabyte line; XA systems only)
- Literals cannot be modified by offsets.

Length of the Comparison

Explicitly define how many bytes of the left side or right side specification should be compared. Certain storage locations have implicit lengths:

- A register or COBOL BLL cell has an implicit length of four bytes
- The storage locations referred to by the CORE keywords MXR, MXS, and TAL have implicit lengths of four bytes
- The implicit length of a COBOL data name is its field length as defined in the DMAP
- The length of a literal is the number of bytes it contains; any length specification is ignored

Define both left side and right side lengths for a packed decimal (COMP-3) comparison. For all other comparisons, define only one length. If you define both lengths, the smaller length is used.

The maximum permissible length for the left side or right side is 16 bytes for packed decimal data and 255 bytes for all other data types.

Literal Formats

For details about specifying literals, see Figurative Constants (see page 349) earlier in this chapter.

CORE Keywords

Specify the following CORE keywords on the left side or right side of the comparison:

Keyword	Function
CMAR	First byte of the EXEC CICS Communications Area for the task
CSA	First byte of the CSA
CURR	Next Assembler instruction to be executed
CWA	First byte of the CWA
CWK	First byte of the COBOL program's Working-storage
DSA	First byte of the program's DSA
ITBE	First byte of the next instruction to be executed
LCL	First byte of the COBOL program's local-storage
MXR	CA InterTest for CICS maximum CICS request counter (implicit length = 4)
MXS	CA InterTest for CICS maximum storage counter (implicit length = 4)
OPFL	First byte of the optional features List
PGM=*	First byte of the monitored program
PREV	Last Assembler instruction executed
R*nn*	A register (*nn* is a decimal from 1 to 15) (implicit length = 4)
TAL	CA InterTest for CICS tally fullword (implicit length = 4)
TERM=*	First byte of the terminal table entry of the current terminal
TGT	First byte of the COBOL TGT for the monitored task
TIOA	First byte of the first TIOA of the task
TWA	First byte of the TWA of the monitored task

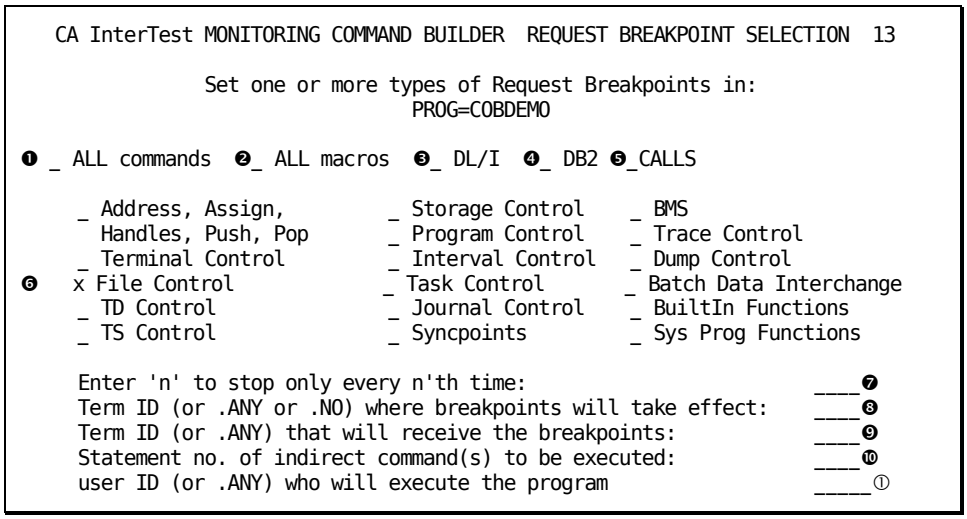
Request Breakpoints

Set request breakpoints to halt a program prior to CICS commands, macros, and other program calls, such as calls to DL/I, DB2, or SQL/DS. Instruct CA InterTest for CICS to halt the program before every CICS command, or a type of CICS command, such as file control or program control commands. Specify that the program halts before specific commands, such as all READ or WRITE commands. Once the program halts, Use all of the CA InterTest for CICS facilities to inspect and modify main storage or auxiliary storage or to set additional options.

To set or remove request breakpoints:

1. Specify function 13, S or R.
2. Specify the program names, transaction codes, or terminal names on the Function Selection Menu, and press Enter.

CA InterTest for CICS displays the Request Breakpoint Selection menu. Enter an **x** next to the options you want to select.



❶

The program halts prior to all CICS commands.

❷

The program halts prior to all CICS macros.

❸

The program halts prior to all DL/I calls.

❹

The program halts prior to all calls to DSNHLI (for DB2 users), or prior to all calls to ARIPRDI (for SQL/DS users).

❺

The program halts prior to calls to software that CA InterTest for CICS has been instructed to recognize at installation time. A second screen opens for you to specify the calls.

❻

If you select a type of CICS command, CA InterTest for CICS displays a second screen. For example, if you select File Control, CA InterTest for CICS displays the File Control screen.

❼

If you do not want the program to halt before every occurrence of a command, specify how often the program should halt. For example, if you specify 3, the breakpoint occurs every third time the program reaches the command. This feature is especially useful when a CICS command occurs within a loop; for example, during a browse.

❽

Specify the ID of the terminal where the program must be executing for breakpoints to take effect. If you leave this field blank, it defaults to your current terminal. Enter .ANY to have breakpoints take effect at all terminals, even when the program executes without a terminal. Enter .NO to have breakpoints take effect only when the program executes without a terminal.

❾

Specify the ID of the terminal to receive breakpoint displays. If you leave this field blank, it defaults to your current terminal. Enter .ANY to have breakpoint screens displayed at the terminal where the program is executing when breakpoints occur.

❿

Specify the first indirect command statement number to take effect at this breakpoint.

①

Specify the CICS user ID of the user (or .ANY) who must execute the program for the breakpoint to take effect.

Note: Your version of this screen might differ slightly, depending on CA InterTest for CICS installation options.

When you have entered all necessary information, press **Enter**. If you entered an x next to a type of command, CA InterTest displays a second screen.

Example

If you selected File Control as the previous screen shows, CA InterTest for CICS displays the following screen.

```
CA InterTest MONITORING COMMAND BUILDER   REQUEST BREAKPOINT SELECTION

      Set  one or more types of File Control commands/macros in:
                          PROG=COBDEMO

        _ All commands            _ All macros

        ➔     x READ
        ➔     x WRITE
              _ REWRITE
              _ DELETE
              _ UNLOCK
              _ STARTBR
              _ READNEXT
              _ READPREV
              _ ENDBR
              _ RESETBR

        Enter 'n' to stop only every n'th time
```

On this screen specify that the program be halted at *all* File Control commands or macros, or select specific File Control commands. In this example, the user specified that program COBDEMO be halted at all READ and WRITE commands.

To remove request breakpoints, use the Monitoring Status display.Alternatively, enter the information *exactly* as you originally defined it.

For more information:

Menus and Displays (see page 35)

Statement Trace Options

Note: These options are unrelated to the backtrace facility.

Set Statement Trace options to save trace information for executed COBOL statements and to capture data values for those statements. Remove the Statement Trace options to delete the trace data.

To set or remove Statement Trace options, follow these steps.

1. Specify function 16, S or R.
2. Specify the program names, transaction codes, or terminal names on the Function Selection Menu, and press **Enter**.

The following Statement Trace Options screen opens. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

```
         CA InterTest MONITORING COMMAND BUILDER – STATEMENT TRACE OPTIONS  16

    SET statement trace options for: PROG=COBDEMO


        Enter X next to each option desired:
             Statement tracing (TRC) _
             Data monitoring   (DM ) _

      Term ID (or .ANY or .NO) where trace options will take effect:   ____

      User ID (or .ANY) who will execute the program:                  _______

  Note: If data monitoring is selected either a specific terminal ID or user ID
        is required.




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

Specify the following options on this screen. The equivalent option used in single-line commands is specified in parentheses.

Statement tracing (TRC)—Instructs CA InterTest for CICS to save trace information for every executed COBOL statement.

Data monitoring (DM)—Instructs CA InterTest for CICS to capture data values for every executed COBOL statement. If data monitoring is selected, statement tracing is automatically activated. In addition, a specific terminal ID or user ID is required.

Term ID (or .ANY or .NO) where trace options will take effect—Specifies that the program be monitored and that Statement Trace options, if specified, take effect only when the program is executed at any terminal (.ANY), no terminal (.NO), or a specific terminal.

User ID (or .ANY) who will execute the program—Specifies that the program be monitored and that Statement Trace options, if specified, take effect only when the program is executed by any user (.ANY) or a specific user (a CICS user ID).

Remove Statement Trace Options

To remove statement trace options, enter an **x** next to the options you want to remove. If you select statement tracing, data monitoring is also removed. Enter the terminal ID and user ID information exactly as you originally defined it.

Replacement Options

Set replacement options to dynamically change the names of CICS resources (programs, files, transient data queues, and temporary storage) specified in CICS CALLS by a monitored program. Remove the replacement options when you want the program to use the resources defined in the program.

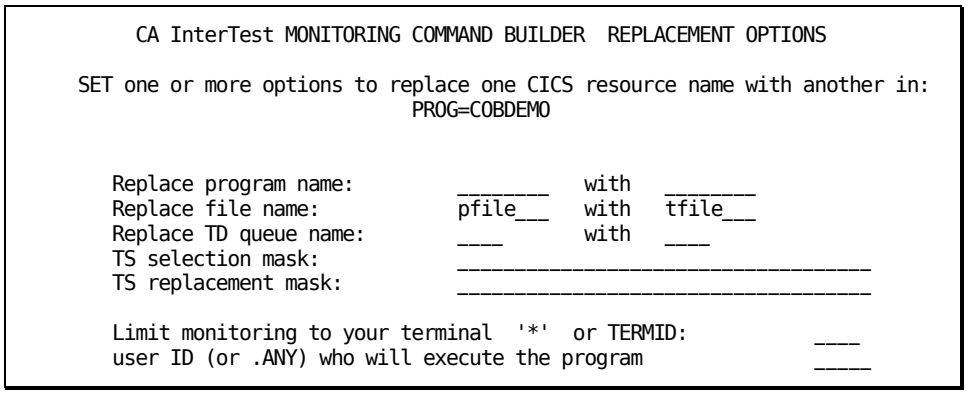
Set Replacement Options

When you specify replacement options for a transaction, the resources are replaced in all its programs. When you specify replacement options for a terminal, the resources are replaced in all programs executing from that terminal.

To set or remove replacement options:

1. Specify function 20, S or R,
2. Specify the program names, transaction codes, or terminal names on the Function Selection Menu, and press **Enter**.

The Replacement Options screen opens. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.



Specify five options on this screen. The equivalent option used in single-line commands is specified in parentheses on the following option list.

Replace program name (RPC)—Allows a program name to be replaced. Specify the original program name and the new name.

Replace file name (RFC)—Allows a file name to be replaced. Specify the original file name and the new name. The sample screen shown previously specifies that program COBDEMO use file TFILE instead of PFILE.

Replace TD queue name (RTD)—Allows a transient data queue name to be replaced. Specify the original TD queue name and the new name.

TS selection mask; TS replacement mask (RTS)—Allows a temporary storage ID to be replaced. Specify the original temporary storage ID in the TS selection mask field, and the new temporary storage ID in the TS replacement mask field. Specify both masks as eight bytes in either character (C'data') or hexadecimal (X'data') format. In the following example, STORAGE2 replaces STORAGE1 in all temporary storage requests issued by the program.

TS selection mask: C'STORAGE1'
TS replacement mask: C'STORAGE2'

Limit monitoring (TON)—Specifies that the program be monitored and that replacement options, if specified, take effect only when the program executes at the current terminal (*), a specific terminal (termid), or without a terminal (.NO).

When you have specified the necessary information, press Enter.

Remove Replacement Options

To remove a replacement option, specify the information exactly as you entered it. To remove all replacement options of a specific type, enter **.ALL** in the appropriate field.

Example

To remove all file replacement options, enter **.ALL** in the Replace file name field.

Protection Options

This section describes protection options.

Set Protection Options

Set options to override the CA InterTest for CICS default protection rules for modifying main storage, the CSA, and load modules. Remove the protection options to reinstate the default protection rules.

Important! Because the protection options can potentially damage your CICS system if misused, exercise caution in specifying them. These options are password protected unless your site has removed this restriction.

To set or remove protection options follow these steps.

1. Specify function 21, S or R**.**
2. Specify the program name on the Function Selection Menu, and press Enter.

   The Protection Options screen shown next opens. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

```
CA InterTest MONITORING COMMAND BUILDER  PROTECTION OPTIONS       21

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


        Bypass storage protection:      From    ________   To      ________
        Unprotect CSA:                  Offset  ____       Length  ___
        Unprotect CWA:                  Offset  ____       Length  ___
        Unprotect main storage area:    Address ________   Length  _____
        Unprotect load module:          Name    ________
        Protect main storage area:      Address ________   Length  _____



        Limit monitoring to your TERMINAL  '*' or TERMID:          ____
        user ID (or .ANY) who will execute the program             _____
```

Specify the following options on this screen. In the descriptions, the command syntax equivalent for each monitoring option is specified in parentheses.

Bypass storage protection (BYP)—Permits a specified section of program code to modify any area of main storage, issue SVC instructions, or issue BALR, BASSM. or BASR 14,15 or 14,14 instructions.

- Specify the beginning and ending program addresses or program offsets. Specify addresses as six- to eight-hexadecimal digits, and offsets as one- to five-hexadecimal digits.
- If you specify .ANY in the From field and an address or offset in the To field, CA InterTest for CICS suspends monitoring when any BALR, BASSM, or BASR 14,15 or 14,14 instruction passes control to the specified location. Monitoring resumes when the routine executed by the instruction returns control to the next byte after the instruction.
- At a breakpoint, specify * in the From field and leave the To field blank to instruct CA InterTest for CICS to bypass the current instruction.

Unprotect CSA (CSA)—Permits a program to modify areas in the CSA. Specify the offset and the length in hexadecimal.

Unprotect CWA (CWA)— Permits a program to modify areas in the CWA. Specify the offset and the length in hexadecimal.

Unprotect main storage area (LET)—Permits a program to modify a designated area of storage. Specify the beginning address as six to eight hexadecimal digits. Specify the length of the area in hexadecimal.

Unprotect load module (LET)—Permits a program to modify a load module.

Protect main storage area (PRO)—Prevents a program from modifying a designated area of storage not protected by CA InterTest. Specify the beginning address as six to eight hexadecimal digits. Specify the length of the area in hexadecimal.

Limit monitoring (TON)—Specifies that the program be monitored and that the protection options, if specified, take effect only when the program executes at the current terminal (*), a specific terminal (termid), or without a terminal (.NO).

Remove Protection Options

To remove protection options, enter the information exactly as you originally defined it. To remove all options of a specific type, enter **.ALL** in the appropriate field.

Example

To remove all bypass storage protection options, enter **.ALL** in that field.

Special Options

This section discusses special options.

Set Special Options

Set special options to alter the CA InterTest for CICS standard monitoring procedures. Remove the special options to reinstate the default monitoring rules.

To set or remove special options, follow these steps.

1. Specify function 22, S or R.
2. Specify the program names, transaction codes, or terminal names on the Function Selection Menu, and press **Enter**.

The following Special Options screen opens. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

```
CA InterTest MONITORING COMMAND BUILDER  SPECIAL OPTIONS      22

     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) _______
        Limit total number of CICS requests  (MXR) _______
        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
```

Specify the following options on this screen. The equivalent option used in single-line commands is specified in parentheses.

Source Listing Breakpoint (SLB)—Specifies that the Source Listing Breakpoint screen be displayed rather than the full breakpoint display screen. For more information, see the chapter "Source Listing Facility."

No file updating (NUP)—Specifies that a monitored program not update any files. This option only affects VSAM and BDAM files; databases are unaffected.

Reentrancy check (RNT)—Prevents a program from modifying its own code.

Follow monitoring (FOL)—Instructs CA InterTest for CICS to continue monitoring a program even if it branches directly to another program (wild branch). All monitoring options remain in effect. Specify the following:

- ON when control passes to a program that does *not* have a PPT entry
- The name of the program when control passes to a program that has a PPT entry
- NOPPT instead of ON to reduce overhead when a program often branches to a program that does not have a PPT entry

  Note: Do not specify this option if control is passed to a program by a LINK or XCTL instruction. Such a program is monitored only if monitoring was specified for it.

Number of times to be monitored (MUS)—Limits the number of times a program should be monitored.

Limit total size of CICS storage (MXS)—Limits the amount of storage, including program storage, a program acquires.

Limit total number of CICS requests (MXR)—Limits the number of requests a transaction issues. This option is useful when a program is in a loop that includes a CICS request.

Setting for Structure Display Format (SDF)—Use this option to override, by program, the global default setting. Specify the following:

- HEX to display data in hexadecimal / character format
- DATA to display data in Structure Display Format

Set local automatic breakpoint (ABP)—Activates the automatic breakpoint facility for a particular program, transaction, or terminal.

Specify the terminal to receive automatic breakpoint displays.

- Asterisk (*) routes the display to the current terminal
- termid routes the display to a particular terminal
- .ANY routes the display to all terminals
- OFF instructs CA InterTest for CICS to abend the task instead of halting it at an automatic breakpoint

Note: This option is useful when you do not want the breakpoint display routed to the terminal where the program is running, or when the display cannot be routed to that terminal (for example, a non-terminal task or a task running on a non-3270 terminal). In those cases, specify the terminal to which breakpoint displays should be routed. The local automatic breakpoint option is also useful in a production environment when the global Automatic Breakpoint facility is disabled and you want to monitor a specific program, transaction, or terminal.

Limit monitoring (TON)—Specifies that the program be monitored and that the special options, if specified, take effect only when the program executes at the current terminal (*), a specific terminal (termid), or without a terminal (.NO).

Remove Special Option

To remove special options, enter an **x** next to the options you want to remove, or enter the information exactly as you originally defined it.

Set and Remove Composite Support

The CA InterTest for CICS composite support feature lets you take advantage of all of the CA InterTest for CICS capabilities when you test and debug composite modules. A *composite module* is a load module defined in the PPT that consists of separately compiled or assembled parts brought together when the module is link-edited. The part of the composite module that receives control from CICS is referred to as the *main program*; the remaining parts are referred to as *subprograms*. The main program and subprograms are written in the same or different languages.

Composite support lets you test and debug a subprogram as if it were a *separate* program. This means you set breakpoints and other monitoring options individually for any subprogram.

CA InterTest for CICS provides *full symbolic support* for the main program and subprograms if you provide that information when the programs are compiled or assembled.

Function 23 on the CNTL Function Selection Menu is one method of accessing the Composite Support screen, a sample screen follows. An alternative method of accessing this screen is from the ITST monitoring menus.

```
         CA InterTest MONITORING COMMAND BUILDER - COMPOSITE SUPPORT      23
                    Define composite support for BIGMOD

   Link-name  Monitor-name  Offset  Length  Language           Comments

_ MAINMOD      BIGMOD        160     78A8  COBOL
_ SUBMOD-A     ASMMOD-A     7CA0     1200  ASSEMBLER
_ SUBMOD-B     ASMMOD-B     8EA0     1001  ASSEMBLER
_ SUBMOD-C     ASMMOD-C     9EA1      93A  ASSEMBLER
_ SUBMOD-D     ________     A8DB      A11  __________ symbolics not available



Add, remove, or change monitor names as needed, and press PF5
CAIN2602 BOTTOM of data
PF1 Help       2           3 End        4 Return     5 Process    6
PF7 Backward   8 Forward   9           10           11           12
```

For more information:

Source Listing Facility (see page 47)
Monitoring Menu Options (see page 227)

Indirect Commands

The CA InterTest for CICS indirect commands facility lets you define a set of commands that execute automatically or *indirectly* at a predetermined location in a monitored COBOL or PL/I program. These commands are invoked from an unconditional, conditional, or request breakpoint.

In effect, indirect commands are executed at breakpoints within your program just as other CA InterTest for CICS line commands issued from the Source Listing display, such as MOVE and GO, are used to alter the flow of your program's logic. The difference is that indirect commands are executed automatically without issuing a breakpoint display. Once the indirect commands are performed, your program continues to execute normally.

Indirect commands save you from having to recompile a program or manually correct an area of code every time it is encountered during a debugging session. With the indirect commands facility, you can:

- 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

  Note: Indirect commands are *not* available for Assembler programs.

When to Use Indirect Commands

Indirect commands are used to correct the following sample program problems:

- A variable is not being initialized during processing. Using indirect commands initialize the variable every time the program is run and continue execution without recompiling.
- A new routine is needed in your program. Insert the code using indirect commands and continue executing your program without recompiling.
- A section of code is bad. To continue executing or testing without recompiling, write and invoke indirect commands to direct CA InterTest for CICS to go around the bad source code or execute a new series of indirect command statements.

Step Overview

For users familiar with the CNTL menus, use Function 24 of the CNTL Function Selection Menu to access the indirect commands facility when you are coding the commands.

For more information:

Breakpoint Activities (see page 135)

Status Display

Determine the monitoring status of one or more programs, transactions, or terminals by following these steps:

1. Specify Function 30**.**
2. Specify the program names, transaction codes, or terminal names on the Function Selection Menu, and press **Enter**.

   If you specify multiple programs, transactions, or terminals, CA InterTest for CICS displays the monitoring reports in the order in which you specified the entries.

Note: The status display accessed from the CNTL menu does not include the same formatting as the monitoring status report discussed in other areas of this guide.

Important! It is advisable to determine the monitoring status of a program, transaction, or terminal before altering it.

For more information:

Menus and Displays (see page 35)

Example

The following screen shows a status report for the program COBDEMO.

```
    CA InterTest r8.5 Activity Report, 11:39 a.m at terminal L905
     CICS Trace for InterTest internals is on.
     No currently monitored tasks found.
 ❶ Automatic BreakPoint feature is globally active.
 ❷ Automatic BreakPoint default terminal is not declared.
 ❸ CKPT  intrval=00:10 hh:mm
 ❹CKPT  done at 11:37  a.m.
❺ COBDEMO    Symbolic file is PROTDEM
               UBP   offset=  +01B44 id=56A7D208 at=#898
                     from=L903 to=L903
               UBP   offset=  +016C2 id=B903CE24 at=#1
    ❻               from=L903 to=L903
               SLB   terminal is L903  Source Listing Breakpoints are off.
                     Listing view profile: Default.
                    Singlestep no less than 002 statements.
          ❼ * Entry not active
             * End of Report *



--------------------------------------------------------------------------------
-
PF1:Top   PF7:Up   PF8 or Enter:Down   PF3 or Clear:End   PF4:Refresh. Page 001
```

This report provides the following information:

❶

The global ABP (Automatic Breakpoint) facility is active. This means monitored programs will receive breakpoint displays when an error triggers an automatic breakpoint.

❷

There is no global ABP terminal.

❸

The checkpoint recording facility is in effect. The interval between checkpoints is 30 minutes.

❹

The last checkpoint occurred at 9:45 a.m.

❺

The next line identifies the program for which the status report was generated as COBDEMO.

❻

The rest of the report indicates the options currently active for COBDEMO. In this example, only Unconditional Breakpoints (UBP) are active for this program. For each breakpoint, the report provides the information shown in the following table.

Field	Information
offset=	The breakpoint location is defined as a hexadecimal offset from the beginning of the program.
id=*hhhhhhhh*	The breakpoint ID number assigned by CA InterTest for CICS.
at=*location*	The symbolic location specified when the breakpoint was set. A number preceded by a # sign refers to a statement number, except for #1, which refers to the first executable instruction.
from=*termid*	The ID of the terminal where the program must be executing for the breakpoint to take effect.
to=*termid*	The ID of the terminal that will receive the breakpoint display.

Note: If you do not see the message, End of Report, the report is too long to fit on one screen. Press **Enter** to display more of the report.

❼

Entry not active. This indicates that CA InterTest for CICS is not currently monitoring the program.

Utility Functions

Use the utility functions to produce CA InterTest for CICS reports and to reset monitoring for a recompiled program.

To select utility functions, specify Function **31** on the Function Selection Menu, and press **Enter**. The following Utility Options screen opens:

```
        CA InterTest MONITORING COMMAND BUILDER  UTILITY OPTIONS       31


ENTER X NEXT TO EACH DESIRED KEYWORD:

FUNCTION                          KEYWORD          OPTIONAL HARDCOPY PRINTOUT

List CA InterTest tables            LIST     _     TD queue       -----
List programs in symbolic file      SYM      _     TD queue       -----
Global monitoring report            INQ      _     TD queue       -----
List program in symbolic file       SYM      _     Program name   -----
New program copy                    NEW      _     Program name   -----
```

Specify the following functions on this screen. The equivalent function used in single-line commands is specified in parentheses.

List CA InterTest tables (LIST)—Requests a list of all the programs, transactions, and terminals that CA InterTest for CICS was instructed to monitor or *not* to monitor. The report also lists the programs CA InterTest for CICS will monitor as a result of the FOL special option.

List programs in symbolic file (SYM)—Requests a list of all the programs in the CA InterTest for CICS symbolic file. For each program, this report lists its date and time of compilation, the number of records it occupies in the file, whether it can be purged, and whether it was compiled with the LISTER option.

Global monitoring report (INQ)—Requests a report of all CA InterTest for CICS monitoring activity.

List one program in symbolic file (SYM)—Requests information for one program in the CA InterTest for CICS symbolic file. It lists the date and time of compilation, the number of records it occupies in the file, whether it can be purged, and whether it was compiled with the LISTER option.

New program copy (NEW)—Resets symbolically specified breakpoints and other monitoring options for a recompiled program. Also resets the entry in the PPT Table to the program's new library address.

Note: The reports produced by the options LIST, SYM, and INQ might be too long to fit on one screen. Press **Enter** to display more of the report. To write a report to a transient data queue, enter the name of the TD queue.

Set SystemWide Options

Set certain options on a system-wide basis. Specify Function 32 on the Function Selection Menu, and press Enter. The following System-Wide Options screen opens:

```
 CA InterTest MONITORING COMMAND BUILDER  SYSTEMWIDE OPTIONS      32

Select one or more of the following actions:

FUNCTION                  KEYWORD ACTION
                                (ON / OFF)

Automatic breakpoint           ABP    ___   Terminal to receive breakpoint ____
Global logging                 GLOG   ___
Systemwide monitoring          ALL    ___


Select one of the following options:

Purge breakpointed tasks       PURGE  ___   Interval in format HHMM        ____
Checkpoint CA InterTest tables CKPT   ___   Interval in format HHMM
```

Specify the following options on this screen. Specify in parentheses the equivalent option used in single-line commands. Enter **ON** to set an option and **OFF** to remove it.

Automatic breakpoint (ABP)— Sets the automatic breakpoint facility for the entire system; the default. Breakpoint displays are routed to the terminal where the program is running. Specify a terminal ID to indicate where breakpoint displays should be routed for non-terminal attached tasks and for tasks executing from non-3270 terminals.

If you specify OFF for the ABP option, you might want to use the Special Options screen (Function 22) to set the local automatic breakpoint option for one or more programs. The local automatic breakpoint option overrides the system automatic breakpoint option.

Global logging (GLOG)—Requests global logging of CA InterTest for CICS activity: most CNTL commands and all changes to main storage.

System-wide monitoring (ALL)—Instructs CA InterTest for CICS to monitor all application programs in the system, except those with the prefix DFH (IBM CICS programs) or IN25 (CA InterTest for CICS programs).

Note: Specify only one of the following two options at one time. To set both facilities, return to the Function Selection Menu and specify Function 32 again.

Purge breakpointed tasks (PURGE)—Instructs CA InterTest for CICS to periodically purge tasks halted at a breakpoint for longer than the specified interval. The first purge occurs when the option is specified. Subsequent purges occur at the end of the interval, specified in hours and minutes in this format *hhmm*. The minimum time interval is 20 minutes.

Checkpoint CA InterTest (CKPT)—Instructs the checkpoint recording facility to record periodically the status of CA InterTest for CICS monitoring so that you can use the restart feature. The first checkpoint occurs when the option is specified. Subsequent checkpoints occur at the end of the interval, specified in hours and minutes in this format *hhmm*. The minimum time interval is 10 minutes.

Note: These options are password protected unless your site has removed this restriction.

Resume Task Execution

When a task is halted at a breakpoint, request that execution of the task resume. Specify Function 33 on the Function Selection Menu and press **Enter**. For detailed instructions, see The Function Selection Menu (see page 337) earlier in this chapter.

The following screen opens:

```
CA InterTest MONITORING COMMAND BUILDER - RESUME TASK EXECUTION     33


 Press Enter without options to resume execution of the breakpointed task  ____
❶
       OR    enter one or more of the following options:


 To RESUME EXECUTION at a different location, enter ONE new location:

      COBOL paragraph, Assembler or PL/1 label:_____________________ ❷
      COBOL or PL/1 statement number: _____  ❸  Offset: _____  ❹
 To MOVE this breakpoint enter ONE new location,  OR  enter X to DISABLE it:_ ❺
        COBOL paragraph, Assembler or PL/1 label:  _____________________________
❻
        COBOL or PL/1 statement number: _____  ❼  Offset: _____  ❽



 To AUTO STEP, enter #n for COBOL verbs or PL/1 STMTS or n for instrucs: ❾
      with a n second WAIT between breakpoints:                          ❿
      to HALT at a CALL, enter X ①  OR after n steps are reached: ②

 To take a DUMP before continuing, enter a 1 to 4 digit dump code:   ③
 To ABEND transaction, enter dump code or XXXX if no dump is desired: ④
```

Complete the previous screen as follows:

- Field ❶contains the number of the task (if any) waiting at a breakpoint that most recently sent a breakpoint display to your terminal. To resume execution of this task, press **Enter**. To resume execution of a different task, enter its task number in this field, and then press Enter.
- Specify one or more of the following options:
- Resume task execution at a new location by specifying the COBOL paragraph name, Assembler or PL/I label in field ❷, the COBOL or PL/I statement number in field ❸, or the COBOL, PL/I, or Assembler offset in field ❹.

  Important! Do not resume execution at a new location in optimized code. In PL/I programs, do not resume execution at a new location if it results in a jump between PROCS/BLOCKS.
- To disable (cancel) the current breakpoint, enter an **X** in field ❺
- Resume task execution and move the current breakpoint to a new location by specifying the new COBOL paragraph name, Assembler label, or PL/I label in field ❻, the COBOL or PL/I statement number in field ❼ or the COBOL, PL/I, or Assembler offset in field ❽.
- To execute a specified number of COBOL verbs, PL/I statements, or instructions before halting the task again, specify the number in field ❾.
- To automatically repeat single-stepping after a specified number of seconds, enter a number from 1 to 59 in field ❿.
- To halt automatic single-stepping at a CALL or CICS command or macro, enter **X** in field ①.
- To halt automatic single-stepping after a specified number of steps, enter a number from 1 to 9999 in Field ②.
- To take a dump before continuing, enter a one- to four-digit dump code in field ③. This code cannot begin with an A.
- To abend the task, enter a one- to four-digit dump code or XXXX (if no dump is desired) in field ④. This code cannot begin with an A. Do not specify this option in conjunction with options 2, 3, 4, 9, or 10.

Example

The following screen shows how to specify task resumption.

This screen specifies that the task:

- Resume at a new location: TASK-NUMBER-FOR-MSG.
- Execute three COBOL verbs before halting again.

```
CA InterTest MONITORING COMMAND BUILDER  RESUME TASK EXECUTION     33

Press ENTER without options to resume execution of the breakpointed task  _____

       OR    enter one or more of the following options:

To RESUME EXECUTION at a different location, enter ONE new location:
     COBOL paragraph, Assembler or PL/1 label:  tasknumberformsg___________
     COBOL or PL/1 statement number: _____      Offset: _____

To MOVE this breakpoint enter ONE new location,  OR  enter X to DISABLE it:
     COBOL paragraph, Assembler or PL/1 label:  _______________________________
     COBOL or PL/1 statement number: _____      Offset: _____
To AUTO STEP, enter #n for COBOL verbs or PL/1 STMTS or n for instrucs:   #3___
with a n second WAIT between breakpoints:____
to HALT at a CALL, enter XOR after n steps are reached:____

To take a DUMP before continuing, enter a 1 to 4 digit dump code:          ____
To ABEND transaction, enter dump code or XXXX if no dump is desired:       ____
```

Chapter 10: Accessing Main Storage: CORE

CORE is the CICS transaction supplied with CA InterTest for CICS that gives you access to main storage areas both before and during a test session. Application programmers typically use CORE interactively to examine and modify areas of main storage while their program is stopped at a breakpoint. Use CORE also to examine and change (subject to password restriction) the contents of CICS tables and control blocks with or without a program at a breakpoint. CORE is independent of the rest of CA InterTest for CICS and you can use it when the CA InterTest for CICS monitor is not active.

The Main Storage Facilities (CORE)

Use CORE in the following ways:

- **Source Listing Facility action characters:** Use a **d** or **m** on the Source Listing Breakpoint screen to display or modify breakpoint-related storage. This method is discussed in the chapter "The Source Listing Facility."
- **Main Storage Menus:** Use Option **3 Main Storage** from the Primary Option menu to access the Main Storage menu for CORE facilities. From a breakpoint, just type **CORE** on the command line to go directly to the Main Storage menu.
- The Main Storage menu lets you access storage related to CICS system, task, and breakpoint areas without using any command syntax. (The menus build CORE commands for you.) Instructions on how to access and complete the menus are discussed later in this chapter.
- **CORE commands:** Once you become familiar with CA InterTest for CICS, use the CORE command formats and advanced options. Enter CORE commands at the bottom of a CORE facility display, in the command line of any breakpoint or Source Listing display, or directly from CICS.

Main Storage Capabilities

Using Main Storage facilities of CA InterTest for CICS, you can do any of these functions:

- Display areas of CICS main storage
  - Program-related areas such as data items or the COMMAREA
  - Task-related areas such as the TCA or TWA
  - System-related areas such as the CSA, CICS tables, or BMS maps
  - Qualified, indexed, or subscripted COBOL data names

- Display a data structure in COBOL DMAP sequence, Assembler DSECT sequence, or CICS areas in DSECT sequence
- Modify areas of CICS main storage by
  - Directly typing over the displayed data
  - Filling in a pre-for matted MOVE command
  - Moving data from one location to another
- Scan backward and forward through CICS main storage for specified data strings
- Load or delete a program load module
- Dynamically acquire CICS main storage
- Dump a specified area to the dump data set
- If you display a COBOL data item when your program is at a breakpoint, CA InterTest for CICS formats the field according to its definition and determines whether the field's current value is valid. For example, if you use CORE to display the contents of a COMP-3 field that contains low-values such as binary zeros, CA InterTest for CICS automatically informs you that the field contains invalid data. In addition, if the field is numeric, such as hexadecimal or packed decimal, CA InterTest for CICS displays its decimal value.

CORE Main Storage Displays

The CORE facility provides two storage display formats: structured and dump. When available, the structured format is the default.

The **structured format** gives field names and displays storage field-by-field, which makes it easy to read and modify. The structured format displays a COBOL data structure in DMAP sequence, an Assembler data structure in DSECT sequence, and common CICS areas in DSECT sequence. You do not have to know or remember field lengths to determine their contents. The basic format of the structured display is:

field **hexadecimal value** **character value**

By contrast, the traditional **dump format** starts from a given point (which you specify) and is a continuous stream of values. You have to remember field lengths to know where one field stops and the next one starts.

Both formats provide similar functional capabilities, including:

- **Full-screen edit** permits you to dynamically modify displayed storage
- **CORE command line** displays the last CORE command executed. Change this command to any CORE or CNTL command.

- **Message line** notifies you of: invalid storage values, the values of numeric data, additional CORE requests queued for display, or if the requested storage area was not found.
- **PF keys** let you scroll, switch the format displayed, access help, and return to a menu or breakpoint.

The Structured Format

There are two versions of the structured format:

- One for COBOL and PL/I
- One for Assembler and CICS data structures

The COBOL and PL/I Structured Display

You obtain a structured display by default when accessing main storage from:

- Source Listing Breakpoint screens
- Main Storage menus

To obtain a structured display when using CORE commands at the bottom of a current storage display, enter the command and press **PF12 Structure**.

Note: If you request a storage display and a structured format is not available, the dump format and a message are displayed.

The following diagram shows the structured format for a COBOL data structure:

```
                CA InterTest for CICS r8.5    - MAIN STORAGE UTILITY - Termid =
 U056

Starting at Address =2370A140  ❶           Hexadecimal            Character
01 TASK-STRUCTURE                |              ❸                |     ❹
 02 TASK-CNTL       ❷            | C3D5E3D3                      | CNTL
 02 TASK-PROTCPF                 | D7D9D6E3 C3D7C640             | PROTCPF
 02 TASK-PROTHLF                 | D7D9D6E3 C8D3C640             | PROTHLF
 02 TASK-SWITCH                  | 40                            |
 02 TASK-SWITCH2                 | F0F0                          | 00
 02 TASK-SWITCH3                 | 40                            |
 02 TASKNUM                      | 000000                        | ...
 02 TASKNUM-CHAR                 | 000000                        | ...
 02 TASK-TEXT                    |                               |
  03 TASK-ID-NO                  | 000F                          | ...
  03 FILLER                      | 40                            |
  03 TASK-MESG                   | E3C8C9E2 40C9E240 C140D4C5    | THIS IS A ME
                                 | E2E2C1C7 C5404040             | SSAGE
  03 FILLER                      | 40                            |
  03 TASK-DATE                   |                               |
 ------------------------------------------------------------------------------
❺PF1 Help       2            3 End        4 Return     5            6 Dump
  PF7 Backward   8 Forward    9 Caps Off  10           11 Redisplay 12 SDF
❻CORE='TASK-STRUCTURE'
❼CAIN0467 AT LAST ENTRY IN STRUCTURE
```

The structured format displays COBOL data in DMAP sequence. The structure of a PL/I variable is presented similarly. Each field is discussed next.

❶

Starting Address of the first item displayed. This is calculated from the contents of the BL and BLL cell and displacement for the requested data name.

❷

DMAP or SYM MAP data levels and names for the requested structure.
A 77-level item is displayed individually; other structured displays begin with the requested data item and include any items below it in the same 01 level.

❸

Hexadecimal display area gives the storage contents of each field named on the left. Each byte is displayed as two hexadecimal digits, with spaces inserted after every four bytes. Up to 12 bytes are displayed per line. Fields longer than 12 bytes have multiple lines of hexadecimal display. This area is unprotected and you can overtype it to dynamically modify the data values displayed.

Note: Data areas defined in the DMAP or SYM MAP with a length of zero (for example, 0F or 0C13) will not display.

❹

Character display area gives the character representation of each displayable byte for the field on the left. Hexadecimal 40s display as spaces; non-displayable bytes are indicated by periods. This area is unprotected and you can overtype it to dynamically modify the data values displayed.

❺

PF keys are listed below the storage display. Each is defined in Structured Displays (see page 383) later in this chapter.

❻

CORE command line displays the CORE command entered or generated from a Source Listing Breakpoint storage request or a CORE menu. To request another structured display, modify the CORE command and press PF12. To request a dump format of a CORE display or to enter a CNTL command, modify the command and press **Enter**.

❼

Message line. Always check the message line on a CORE display. This message states that the end of the requested data structure has been reached. (Page backward to display the beginning of a data structure or page forward when you are not at the end of the structure.) However, this message line also tells you if the requested area could not be found (the display shows the last area pointed to by CORE or the CSA), or if a COBOL field or PL/I variable displayed from a breakpoint contains valid data. An error message indicates when a field is invalid.

The Assembler and CICS Areas Structured Display

You obtain a structured display by default when accessing Assembler main storage from:

- Source Listing Breakpoint screens
- Main Storage menus

To get a structured display when executing a CORE command from an existing CORE display, type the command and press **PF12** instead of Enter.

Note: If a structured format is not available for the requested area or data, the dump format and a message display.

The following diagram gives the structured format for Assembler and CICS areas. The fields are displayed in DSECT sequence.

```
 CA InterTest for CICS r8.5    - MAIN STORAGE UTILITY - Termid = U056
   ❶                    ❷      ❸               ❹                   ❺
Starting at Address =  100CE8  Disp         HEXADECIMAL            CHARACTER
TASKNUM                         189 | 000000                      | ...
TSTEXT                          18C | 00000000 00000000 00000000 | ............
                                    | 00000000 00000000 00000000 | ............
                                    | 00000000 00000000          | ........
MSGNAME                         1AC | 00000000                  | ....
MSGLEN                          1B0 | 01D5                      | .N
RECRBA                          1B2 | 00000000 0000             | ......
RECPONT                         1B8 | 00000000                  | ....
                                1BC | 00                        | .
DMAP04AS                        1BC |                           |
                                1BC | 00000000 00000000 00000000 | ............
RECOUT1L                        1C8 | 0000                      | ..
RECOUT1F                        1CA | 00                        | .
RECOUT1A                        1CA | 00                        | .
RECOUT1I                        1CB | 00                        | .
RECOUT1O                        1CB | 00000000 00000000 00000000 | ............
 -------------------------------------------------------------------------------
❻PF1 Help       2            3 End        4 Return     5            6 Dump

 PF7 Backward   8 Forward    9 Caps Off  10           11 Redisplay 12 SDF
❼CORE='R13.TASKNUM'
```

❶

Individual field names as defined in the DSECT.

❷

CICS address of each field.

❸

Displacement column is the field's relative hexadecimal displacement within the DSECT.

❹

Hexadecimal display area gives the storage contents of each field named on the left. Each byte is displayed as two hexadecimal digits with spaces inserted after every four bytes. Up to 16 bytes display per line. Fields longer than 16 bytes have multiple lines of hexadecimal display. This area is unprotected and you can overtype it to dynamically modify the data values displayed.

❺

Character display area gives the character representation of each displayable byte for the field on the left. Hexadecimal 40s display as spaces; non-displayable bytes are indicated by periods. Overtype this area to dynamically modify the data values displayed.

❻

PF keys are listed below the storage display. Each is defined in Structured Displays (see page 383) later in this chapter.

❼

CORE command line displays the CORE command entered or generated from a Source Listing Breakpoint storage request or a CORE menu. To request another structured display, modify the CORE command and press **PF12**. To request a dump format of a CORE display or to enter a CNTL command, modify the command and press **Enter**.

❽

Message line. Always check the message line on a CORE display. If the requested area could not be found (the display shows the last area pointed to by CORE or the CSA), a message on this line states this.

Use Another Program's Data Definitions (USE Option)

Structured display is normally used to format an application's storage area with its own data definitions. However, format *any* CICS storage area with *any* program's data definitions. To use another program's data definitions, add one of the following specifications as the last element in a CORE command:

=USE=*progname*.*dataname*

or

=USE=*.*dataname*

progname

Specifies the name of the program that contains the structured information.

dataname

Specifies the name of the data item used as the starting point for the structured display.

*****

Specifies the program currently stopped at a breakpoint.

Examples

Each command is preceded by an explanation of how it works.

- To display the area pointed to by register 1 according to the data definition of COMAREA in the program stopped at a breakpoint, enter:

  CORE@R1=USE=*.COMAREA

- To display the first FWA area according to the data definition of RECORD1, which is in the TESTMAST program, enter:

  ```
  CORE=FWA=USE=TESTMAST.RECORD1
  ```

Structured Displays of DSECTs in PRINT OFF Areas

To get a structured display for a DSECT in a PRINT OFF area:

1. Create an Assembler job containing your common DSECT layouts and use the CA InterTest for CICS post-processor step. To save space in the CA InterTest for CICS Symbolic File:

   - Do not specify the LISTER option.
   - Do not link-edit the module.
   - No PPT entry is necessary.

   The module name must be unique in the CA InterTest for CICS symbolic file, or create one program containing all the common DSECTs and have them printed in the Assembler listing.

2. To reference the module created in Step 1, add the following USE option to your CORE commands:

   ```
   CORE=USE=modulename
   ```

   This command instructs CORE to use the named module instead of the program stopped at a breakpoint when interpreting any symbolic names in the command.

Example

To get a structured display of the USER TCA, which is the DFHTCADS DSECT whose address is in register 12 and is saved in the module TESTMAST, enter:

```
CORE='R12.DFHTCADS'=USE=TESTMAST
```

Structured Displays of Common CICS Areas

The common CICS areas that display in a structured format (Assembler DSECT layout) depend on your CA InterTest for CICS installation. The default system areas available in a structured format are listed in the following table:

CORE Keyword	CICS Area
CLOT	CICS Life-Of-Task Block (z/OS only)
CSA	Common System Area
DCA	Destination Control Area
EIB	EXEC Interface Block

CORE Keyword	CICS Area
EIS	EXEC Interface Structure
OPFL	Optional Features List
RDIIN	Application Invocation Parameter List (z/OS only)
SIT	System Initialization Table
SQLCA	SQL Communication Area (z/OS only)
TACB	Transaction Abend Control Block
TCA	Transaction Control Area
TCT	Transaction Control Table

Note: To add or subtract areas from the previous list, see *the Installation Guide*.

To get a structured display of system areas, use the System Areas menu or enter a **CORE** command, and press **PF12**.

Structured Displays PF Keys

The following table shows the PF keys and their associated functions:

PK Key	Function
PF1 Help	Initiates the CA InterTest for CICS online help facility.
PF2	Unassigned
PF3 End	Exits the current menu or CORE display. If at a breakpoint, returns to the breakpoint display. Clear performs the same function.
PF4 Return	Returns to the Primary Option menu or exits to CICS.
PF5	Unassigned
PF6	This PF key lets the cursor be placed on a valid storage pointer within the hexadecimal display area and the storage pointed to by the pointer will be displayed when this key is pressed. This is referred to as _point and shoot‘ capability and allows for faster navigation through storage without having to key storage addresses on the CORE command line.
PF7 Backward	Scrolls the current CORE display one page backward.

PK Key	Function
PF8 Forward	Scrolls the current CORE display one page forward.
PF9 Caps Off or Caps On	Sets the mode of translation when changing data in the character area of the display. When Caps Off displays, all character data on the line is translated to uppercase. When Caps On displays, all character data is accepted as typed. To change from one mode to the other, simply press PF9. Note: Translation to uppercase will not occur if an existing lowercase character on that line would be affected; a message is issued instead.
PF10	Varies according to what displays.
PF11 Redisplay	Redisplays the current screen with only the contents of storage being refreshed.
PF12 Structure	Displays requested fields in a structured format.

The Dump Format

The dump format is similar to a transaction dump printout, with a hexadecimal display area on the left and a character display area on the right. Obtain a dump format in the following ways:

- Use the Source Listing Breakpoint screen to display or modify a data item or area **without** a structure such as object code.
- Press **Enter** when completing a CORE menu or executing a CORE command.

The following display shows the dump format display of the CICS CSA (Common System Area).

```
❶           CA InterTest  MAIN STORAGE UTILITY  Termid = X520     ❹   ❺
Offset           ❷                              ❸                Address Task
+   0  00000000 00104FA8 001056F0 504CC492  * ......y....0&<Dk *  55EAE0  921
+  10  00BA8000 004CEA90 00203004 9054A00C  * .....<.......... *  55EAF0  ❻
+  20  0000001B 001FD48A 405499DA 4054D400  * ......M. .r. .M. *  55EB00  Page
+  30  0054A9DA 001FD480 00203004 004EA410  * ..z...M......+u. *  55EB10  +000
+  40  004CE800 0021F140 0010999C 0021F140  * .<Y...1 ..r...1  *  55EB20
+  50  1858387F 001FE6E0 03E80100 00000000  * ..."..W..Y...... *  55EB30  CSA
+  60  00683F03 00012C00 007F9308 000095BC  * .........."l...n. *  55EB40
+  70  0000A000 00105800 00569000 0087362F  * ............g.. *  55EB50
+  80  00560190 E0FFFFBE 0000004B 00561438  * ............... *  55EB60
+  90  0000007B 00000000 00382030 D438C616  * ...#........M.F. *  55EB70
+  A0  00000000 001FE580 001FE240 001FE190  * ......V...S .... *  55EB80
+  B0  001FE300 0055F1B8 0055F388 C500FF00  * ..T...1...3hE... *  55EB90
+  C0  000A0004 00000400 0055EEE0 00000000  * ............... *  55EBA0
+  D0  00000000 00000000 007F9308 00000000  * .........."l..... *  55EBB0
+  E0  40558718 004CC010 004BF9C0 00551CB0  *  .g..<....9..... *  55EBC0
+  F0  00492556 004CEF30 004BA0B8 004AC100  * .....<.......cA. *  55EBD0
+ 100  004AEE94 0049E10A 0048AF74 0047ABD0  * .c.m........... *  55EBE0
_____________________________________________________________________________
❼
 PF1 Help       2           3 End        4 Return     5          6 Dump
 PF7 Backward   8 Forward   9 Caps Off  10           11         12 Structure
```

❶

Offset gives the hexadecimal displacement of the first byte of data on that line relative to the first byte of data on page number 0. Page number 0 is the first screen of a CORE display.

❷

Hexadecimal display area shows each byte of data as two hexadecimal digits. Data is displayed 16 bytes per line, with spaces inserted after every four bytes (eight hexadecimal digits). The maximum size of a single CORE display is 17 lines, or 272 (hexadecimal 110) bytes of data.

❸

Character display area gives the character representation for each displayable byte. Hexadecimal 40s display as spaces; non-displayable bytes are indicated by periods. The columns of asterisks are not part of the data.

❹

Address is the CICS hexadecimal address of the first byte of the line.

❺

Task gives the task number of the breakpointed task for breakpoint-related storage displays. For task-related storage displays, gives the task number specified. (The default for task-related storage is also the task number of the breakpointed task.)

❻

Page. Page numbers are relative to the initial screen display. Paging forward with **PF8** increases the page count; paging backward with **PF7** decreases the page count.

❼

The area below the display area includes the following items:

- PF Key list.
- The command line in the Dump Format of CICS CSA displays the last CORE command entered or generated for the current display. To request a structured format of the current CORE display, press **PF12**. To request another CORE display or to enter a CNTL command, modify the command and press **Enter** (for a dump format or a CNTL command) or **PF12** (for a structured format).
- The message line follows the command line and contains valuable information. Always read the message line on a CORE display.
- If CA InterTest for CICS could not locate a requested display area, the error message indicates that the requested area was not found. When this occurs, CA InterTest for CICS displays the dump format of the last or default CORE pointer location. The default area is the CSA.

Some messages can help you troubleshoot problems in storage.

Example

Whenever you are stopped at a breakpoint and request a COBOL storage display, CA InterTest for CICS automatically checks that the fields in the display contain valid data (according to their definitions). An error message tells you which fields are invalid and why.

Dump Display PF Keys

The following table explains what each key on the Dump Display does.

PF Key	Function
PF1 Help	Initiates the CA InterTest for CICS online help facility.
PF2	Unassigned
PF3 End	Terminates the current menu or CORE display. If at a breakpoint, returns to the breakpoint display. Clear performs the same function.
PF4 Return	Returns to the Primary Option menu or exits to CICS.
PF5	Unassigned
PF6	Unassigned This PF key lets the cursor be placed on a valid storage pointer within the hexadecimal display area and the storage pointed to by the pointer will be displayed when this key is pressed. This is referred to as 'point and shoot' capability and allows for faster navigation through storage without having to key storage addresses on the CORE command line.
PF7 Backward	Scrolls the current CORE display one page backward.
PF8 Forward	Scrolls the current CORE display one page forward.
PF9 Caps Off or Caps On	Sets the mode of translation when changing data in the character area of the display. When Caps Off displays, all character data on the line is translated to uppercase. When Caps On displays, all character data is accepted as typed. To change from one mode to the other, simply press PF9. **Note:** Translation to uppercase will not occur if an existing lowercase character on that line would be affected; a message is issued instead.
PF10	Unassigned
PF11	Unassigned
PF12 Structure	Displays requested fields in a structured format.

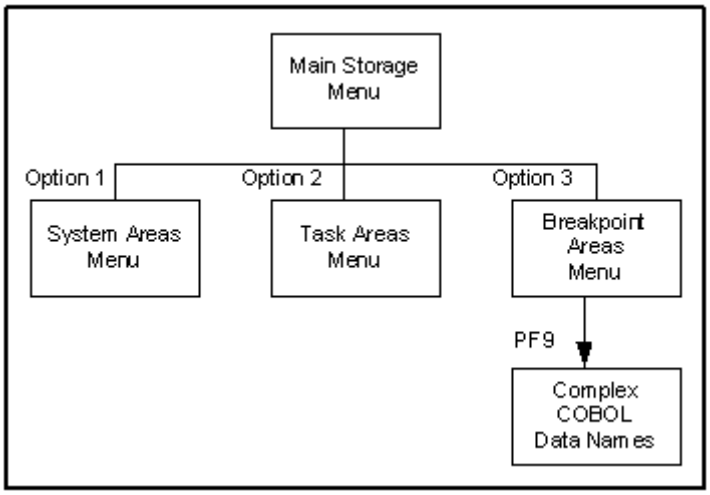
Note: Use **PF13** to **PF24** to perform the same functions as **PF1** to **PF12**.

Access Storage from the Menus

Access the Main Storage (CORE) menus as follows:

- From a Source Listing breakpoint or display, enter **CORE** on the top command line and press Enter.
- From the Primary Option menu, enter Option **3 Main Storage**.

CA InterTest for CICS displays the Main Storage Menu. The submenus available from the Main Storage Menu are shown next.



After completing a CORE menu, press Enter to get a structured display of main storage. From a structured display, press Enter if you want the dump format. The two formats are discussed in CORE Main Storage Displays (see page 376) earlier in this chapter.

The following table summarizes what you can do from each Main Storage menu.

Main Storage Option	Functions
1. System areas	Display, change, or scan: CICS system areas (CSA, OPFL, TCT); individual table entries in the TCT; CICS addresses. Display, load, or delete program load modules and BMS maps. Change functions can be password-protected
2. Task areas	Get storage for a task. Display, change, or scan storage for these task-related CICS areas: CLOT, DWE, JCA, LLA, PLCB, PTA, SQLCA, SQLRCODE, RDIIN, SMX, STCA, TACB, TASKENT, TCA, TIOA, TSIOA, TWA, USER, XMTXN.

Main Storage Option	Functions
3. Breakpoint areas	Display, change, scan, or verify the following breakpoint-related data: data items, registers, or program locations; special elements (includes Breakpoint Message, COBOL BLL cells, COMMAREA, Facility Control Area, COBOL TGT, Saved Screen Image, COBOL Working-storage, COBOL Local-Storage, Exec Interface Block, TCT User Area); access the Complex data menu to display qualified, indexed, and subscripted COBOL data names when at a breakpoint.

Note: Use additional keywords to access other areas. See the help file (see page 543) for an expanded list.

PF Key Use at a CORE Menu

To access another CORE menu or exit from a CORE menu, use any of the PF keys listed in the following table.

PF Key	Function
PF1 Help	Initiates the CA InterTest for CICS online help facility.
PF2	Unassigned
PF3 End *	Exits the CORE menu or display. When stopped at a breakpoint, use PF3 or Clear to restore the current breakpoint display. You might need to use the key more than once.
PF4 Return	Returns to the Primary Option menu.
PF5	Unassigned
PF6	Unassigned
PF7	Unassigned
PF8	Unassigned
PF9 Complex Data	From the Breakpoint menu, displays the menu for viewing qualified, indexed, and subscripted COBOL data; from the System and Task menus, PF9 is unassigned.
PF10	Unassigned
PF11	Unassigned
PF12 Structure	Displays requested fields in a structured format.

Note: PF13 to PF24 perform the same functions as PF1 to PF12, respectively.

Breakpoint-Related Areas Menu

When your task is stopped at a breakpoint, you can display, change, or scan storage relating to the current program or task. From any breakpoint, enter **CORE** on the command line. From the Main Storage menu, select **3 Breakpoint areas**.

CA InterTest for CICS responds by displaying one of three Breakpoint menus, depending on whether your program is written in COBOL, Assembler, or PL/I. The COBOL menu is shown following. Menus used for PL/I and Assembler programs are similar and are described next.

```
    CA InterTest CORE COMMAND BUILDER  BREAKPOINTRELATED AREAS  (CORE=Bkpt)

Specify area to be displayed, changed or moved:              Task number: 00136


  COBOL name: _______________________________

  Special element: ________  (Enter highlighted keyword)
    SSCR Saved screen image   BMSG Breakpoint message       CMAR Commarea
    FCAR Facility cntl area   EIB  Exec interface block     TUAR TCT user area
    BLLS COBOL BLL CELLS      CWK  COBOL WORKING STORAGE    TGT  COBOL TGT
    LCL  COBOL local storage  DSA  COBOL DSA

SCAN VALUE: ____________________________________________          DATA  FORMATS
 SCAN RANGE: ______            B to scan backwards: _
                                                             | P' ' = Packed |
To VERIFY and/or CHANGE Data:                                | X' ' = Hex    |
Existing data: ____________________________________          | C' ' = Char   |
     New data: ____________________________________

MOVE From _______________________________    To   _______________________________


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

Use this menu to do the following tasks:

- Display storage
- Dynamically modify storage
- Verify the contents of storage
- Search storage (scan) for a data string

All storage areas relate to your current task, which is indicated by the number in the upper-right corner of the screen. Overtype the task number to access storage relating to another task stopped at a breakpoint.

Display Storage

Use the COBOL name or Special element field to specify the starting location (the first byte) of the area to display.

Note: To display qualified, indexed, or subscripted COBOL data names, do not enter a COBOL name on this screen. Instead, press **PF9** to access the CORE=Complex COBOL Data Names screen.

COBOL Name

Specify a COBOL variable, a data item, or a paragraph name. If you enter a COBOL paragraph name, CA InterTest for CICS displays the program code at that location.

Special Element

Specify one of the CORE keywords highlighted on the screen to display a special element. Most special elements are familiar storage areas, such as the COMMAREA and COBOL WORKING-STORAGE. The SSCR (Saved screen image) is a special CA InterTest for CICS area that lets you view the last screen displayed by your program before it was stopped at the current breakpoint.

After specifying a COBOL name or Special element, press Enter to obtain a structured display in COBOL DMAP sequence. To get a dump display, press **Enter** a second time.

Examples

To display the contents of the data item TASKNUM in a structured format, specify the following, and press **PF12**.

```
COBOL name: TASKNUM
```

To display the COBOL Working-Storage area for the task currently stopped at a breakpoint, specify the following, and press **PF12** for a structured display.

```
Special element: CWK
```

To display the EXEC Interface Block (EIB) in a COBOL DMAP sequence, specify the following, and press **PF12**.

```
Special element: EIB
```

Display Qualified, Indexed, or Subscripted COBOL Data Names

To display qualified, indexed, or subscripted COBOL data names, **press PF9** while you are viewing the Breakpoint-Related Areas menu (CORE=Bkpt). CA InterTest for CICS displays the Complex COBOL Data Names screen. Parentheses and commas are **not** used in the COBOL Complex data menu.

The following example requests a display of TASKNUM OF TASK-TEXT.

```
  CA InterTest CORE COMMAND BUILDER  COMPLEX COBOL DATA NAMES

 Specify area to be Indexed or Subscripted with or without Qualification:

   Task number:       00136

❶  Data  name:        tasknum_____________________
      OF
❷ Data  name:        tasktext____________________

❸  Index or Subscript1: _____________________________
    Index or Subscript2: _____________________________
    Index or Subscript3: _____________________________



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

Note: An alternative method of displaying complex data names is to use a **d** and the cursor on the Source Listing Breakpoint display. This method saves you from having to type long names. To get a Source Listing Breakpoint display, press **PF4.**

To display the contents of a qualified data name, specify:

❶

Field name in the top Data name field.

❷

Group name in the OF Data name field.

To display the contents of an indexed or subscripted data name, specify:

❶

Field name in the top Data name field.

❷

Qualifying group name in the OF Data name field, if applicable.

❸

Indexes or subscripts in the bottom three fields. Specify one index or subscript per field. Subscripts are designated either numerically or as data names.

Note: You cannot mix indexes and subscripts.

For more information:

Source Listing Facility (see page 47)

Examples

These examples show how to complete the Complex COBOL Data Names menu.

- To display the subscripted item LEVEL-NAME (SUB-1, SUB-2) where the value of SUB-2 is 30, enter:

```
Data name: level-name
OF Data name:
Index or subscript1: sub-1
Index or subscript2: 30
Index or subscript3:
```

- To display the indexed item TABLE-ITEM (INDX-1, INDX-2, INDX-3), enter:

```
Data name: table-item
OF Data name:
Index or subscript1: indx-1
Index or subscript2: indx-1
Index or subscript3: indx-3
```

Dynamically Modify Storage

To modify storage, use the MOVE line on the Breakpoint-Related Areas menu.

The following example gives TASKNUM a value of zero through the entries shown in the MOVE From and To fields.

Note: CA InterTest for CICS specifies the field's length and data type.

```
   CA InterTest CORE COMMAND BUILDER  BREAKPOINTRELATED AREAS  (CORE=Bkpt)

Specify area to be displayed, changed or moved:              Task number: 00136

  COBOL name: _______________________________

  Special element: ________  (Enter highlighted keyword)
    SSCR Saved screen image   BMSG Breakpoint message       CMAR Commarea
    FCAR Facility cntl area   EIB  Exec interface block     TUAR TCT user area
    BLLS COBOL BLL CELLS      CWK  COBOL WORKING STORAGE    TGT  COBOL TGT
    LCL  COBOL local storage  DSA  COBOL DSA

SCAN VALUE: ________________________________________            DATA  FORMATS
SCAN RANGE: ______            B to scan backwards: _
                                                              | P' ' = Packed |
To VERIFY and/or CHANGE Data:                                 | X' ' = Hex    |
Existing data: ____________________________________           | C' ' = Char   |
     New data: ____________________________________

➔  MOVE From zeros_____________________  To   tasknum_____________________


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

For COBOL programmers, using this menu is the easiest way to modify storage because it requires no knowledge of data formats, hexadecimal values, or internal storage representations.

Define Literals

The previous screen shows how to move a literal keyword into the To field.

- Numeric literals are decimal strings of up to 18 digits, optionally preceded by a plus or minus sign.
- Non-numeric literals are any string of EBCDIC characters enclosed in single quotes.
- The character string cannot contain a quotation mark.

To specify a keyword in the From field, use any of the keywords listed in the following table.

From Field Keyword	Actual Value
ZERO(S)	0
SPACE(S)	One or more blanks (X'40')
HIGH-VALUE(S)	One or more occurrences of the hexadecimal string FF
LOW-VALUE(S)	One or more occurrences of the hexadecimal string 00
QUOTE(S)	One or more occurrences of the hexadecimal string 7D (the EBCDIC representation of a quotation mark)

ALL 'xx' or ALL *nn* fills the receiving field with a character string or a decimal digit.

Examples

The following examples demonstrate how to fill a field with a literal.

- To fill the data item FLAG with high-values, enter:

  ```
  From  HIGH-VALUES      To  FLAG
  ```

- To fill COUNTER with low-values (binary zeros), enter:

  ```
  From  LOW-VALUES       To  COUNTER
  ```

- To move the three-byte character string ABC into the DEPT field, enter:

  ```
  From  'ABC'            To  DEPT
  ```

- To fill the field INITIALS with the character string SF, enter:

  ```
  From  ALL 'SF'         To  INITIALS
  ```

Other Ways of Changing Storage

Full-screen edit provides an alternate method of changing storage. It enables you to change any displayed data by overtyping the hexadecimal or character portions of displayed data with new data. Overwrite any number of displayed bytes whenever the bytes are contiguous or separate.

- Overtype hexadecimal display fields with the characters A-F and 0-9.
- Overtype character fields with any character.
- If you overtype the same data on both the hexadecimal and character display fields, the change made in the character field is done first.
- When overtyping the character portion, the data is accepted as entered or converted to uppercase, depending on which mode the CORE transaction is in.

- Uppercase translation occurs when the PF9 key says Caps Off. No translation occurs when it says Caps On. To change the mode, press the **PF9** key before entering any data.
- If you enter commands together with changes, the changes are made before the commands are executed.

Use Verify and Change Data

Another way of changing storage is to specify the starting byte of the area you want to change using the COBOL name or Special element field. Enter the new data in the New data field. It is a good idea, though not required, to enter the current data in the Existing data field. If you do, CORE verifies the field contents before making the data change. If the verification fails, CA InterTest for CICS informs you, and the change is not made.

Specify existing and new data as follows: as a character string (C'...'), hexadecimal value (X'...'), packed data (P'value'), a binary fullword (F'value'), or a binary halfword (H'value') where the value is a decimal number optionally preceded by a plus (+) or minus (-) sign. If no sign is specified, it defaults to +.

Example

1. The entries on the following screen show how to change the contents of the CUST-NAME field from JIMMY to RONALD.

```
    CA InterTest CORE COMMAND BUILDER - BREAKPOINT-RELATED AREAS  (CORE=Bkpt)

Specify area to be displayed, changed or moved:            Task number: 00081

  COBOL name: CUST-NAME____________________

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

SCAN VALUE: ____________________________________________        DATA  FORMATS
SCAN RANGE: ______            B to scan backwards: _          ----------------
                                                             | P' ' = Packed |
To VERIFY and/or CHANGE Data:                                | X' ' = Hex    |
Existing data: C'JIMMY'_____________________________         | C' ' = Char   |
     New data: C'RONALD'____________________________         ----------------

MOVE From _________________________________    To   _________________________________


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

   CA InterTest for CICS performs the requested change only if it verifies that the current contents of CUST-NAME are JIMMY. If CA InterTest for CICS does not find JIMMY in CUST-NAME, it will not perform the change and displays a message to that effect.

2. To change the contents of CUST-NAME to RONALD regardless of its current contents, enter nothing in the Existing data field, and in the New data field enter:

   Existing Data:
   New data: C'RONALD'

3. To confirm the contents of CUST-NAME without changing them, enter nothing in the New data field, and in the Existing data field enter:
   Existing data: C'JIMMY'
   New Data:

4. CA InterTest for CICS informs you whether the prescribed data was found in the designated field.
5. Combine data formats in a single expression.

Example

To specify New data as the character string KEY-CODE followed by two bytes of binary zeros, enter:

Existing data:
New data: C'KEY-CODE'X'0000'

Search for Data

The Breakpoint-Related Areas menu also helps you search for data, as shown in the following screen.

```
CA InterTest CORE COMMAND BUILDER  BREAKPOINTRELATED AREAS  (CORE=Bkpt)

  Specify area to be displayed, changed or moved:           Task number: 00136

  ❶ COBOL name: TABLE1_____________________________

  ❷  Special element: ________  (Enter highlighted keyword)
      SSCR Saved screen image   BMSG Breakpoint message     CMAR Commarea
      FCAR Facility cntl area   EIB  Exec interface block   TUAR TCT user area
      BLLS COBOL BLL CELLS      CWK  COBOL WORKING STORAGE  TGT  COBOL TGT
      LCL  COBOL local storage  DSA  COBOL DSA

❸ SCAN VALUE: C'STEVE F'_____________________________      DATA  FORMATS
❹SCAN RANGE: 1000__            B to scan backwards:   ❺
                                                          | P' ' = Packed |
  To VERIFY and/or CHANGE Data:                           | X' ' = Hex    |
  Existing data: ___________________________________      | C' ' = Char   |
       New data: ___________________________________

  MOVE From _________________________________  To  _________________________________


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

To search for data, perform the following steps:

1. Specify the starting address of the area to search by making an appropriate entry in field ❶ or❷.
2. Enter the data string for which you want to search (in field ❸). Specify the data in character (C'character string') or hexadecimal (X'hex data') format. Combined formats are allowed, such as:

   C`'KEY-CODE'X'FFFF'

3. Specify the maximum number of bytes of storage you want searched (in field ❹). Specify it in hexadecimal format or as a decimal number followed by the letter T.
4. Search backward from the starting location, enter **B** in field ❺. If you leave this field blank, the default is to search forward.

Example

The previous screen is set up to search for the character string STEVE F, beginning at the program location TABLE1, and searching for 1000 hexadecimal bytes at most.

Assembler Version of the Breakpoint Areas Menu

Access the CORE Breakpoint-Related Areas menu for an Assembler program stopped at a breakpoint. From any breakpoint, enter **CORE** on the command line. From the Main Storage menu, select **3 Breakpoint areas**.

```
       CA InterTest CORE COMMAND BUILDER  BREAKPOINTRELATED AREAS  (CORE=Bkpt)


Specify area to be DISPLAYED, CHANGED OR MOVED:              Task number: 00676
                  ❶                              ❷
  Assembler base: ________      Assembler Label: ________________________________
       Register # __ ❸              Argument # __ ❹

  Special element: ________ (Enter highlighted keyword)
    SSCR Saved screen image   BMSG Breakpoint message       CMAR Commarea
    FCAR Facility cntl area   EIB  Exec interface block     TUAR TCT user area

Optional offset: ____________________________________

Scan value: ____________________________________           Data  Formats
Scan range: ______            B to scan backwards: _
                                                          | P' ' = Packed |
To VERIFY and/or CHANGE Data:                             | X' ' = Hex    |
Existing data: ____________________________________       | C' ' = Char   |
     New data: ____________________________________


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

Use the CORE=Bkpt Menu screen to display or change storage relating to the task currently stopped at a breakpoint. All fields, except the numbered ones, operate the same as for the CORE=Bkpt menu displayed for a COBOL program stopped at a breakpoint.

To display the contents of a register, a program location, or a data location, perform the following steps:

1. Make the appropriate entries in fields ❶, ❷, and ❸ as described next.
2. To display the program code at a particular location, enter its label in field ❷.
3. To display the contents of a data field, specify its location by entering the name of the base register in field ❶, and its displacement (from the address in the base register) in field ❷.
4. To display the contents of register *n* (a decimal number from 0 to 15), enter the desired register number in field ❸.
5. To display an argument on your program's Argument List, enter a decimal number from 0 to 99 in field ❹.

Examples

To display the program code at SCANLOOP, enter:

```
Assembler Label: SCANLOOP
```

To display the area at displacement TIOADBA from the address in the register equated to TIOABAR, enter:

```
Assembler base: TIOABAR    Assembler Label: TIOADBA
```

To display the contents of register 8, enter:

```
Register #: 8
```

Note: The address of the Argument List must be in register 1.

To display the second argument in the program's Argument List, enter:

```
Argument: 2
```

PL/I Version of the Breakpoint Areas Menu

Access the CORE Breakpoint-Related Areas menu for a PL/I program stopped at a breakpoint by typing the command: **CORE** on any Breakpoint display. From the Main Storage menu, select Option **3 Breakpoint areas** to display the following menu.

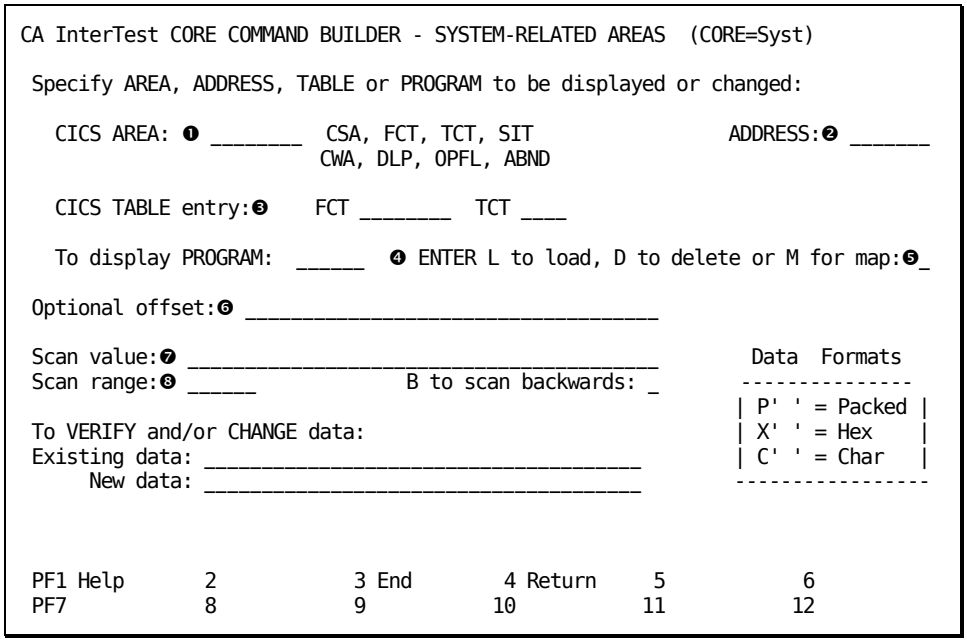
```
              CA InterTest CORE COMMAND BUILDER  BREAKPOINTRELATED AREAS  (CORE=
Bkpt)

Specify area to be displayed, changed or moved:              Task number: 01699

  PL/1 data name: _____________________________

  Special element: ________  (Enter highlighted keyword)
    SSCR Saved screen image   BMSG Breakpoint message       CMAR Commarea
    FCAR Facility cntl area   EIB  Exec interface block     TUAR TCT user area

Optional offset: _____________________________________

Scan value: _____________________________________            Data  Formats
Scan range: ______            B to scan backwards: _
                                                           | P' ' = Packed |
To VERIFY and/or CHANGE Data:                              | X' ' = Hex    |
Existing data: _____________________________________       | C' ' = Char   |
     New data: _____________________________________


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

Use this screen to display or change storage relating to the task currently stopped at a breakpoint. All fields operate the same as for the CORE=Bkpt menu displayed for a COBOL program stopped at a breakpoint, except you enter a PL/I data name in the first field instead of a COBOL data name. For a description of this screen, see Breakpoint-Related Areas menu earlier in this chapter.

System-Related Areas Menu

To display the System-Related Areas menu, enter **CORE** on a clear screen or from the command line of any source listing display or breakpoint. This displays the Main Storage menu. Select option **1 System areas.**

Use the System-Related Areas menu to display or change (subject to password protection) areas of storage relating to CICS. You can also load and delete program modules and search for data strings.



Display System-Related Areas

The System-Related Areas menu lets you display the following, either from the beginning or from a specified offset or scan value:

- CICS area listed on the menu (field ❶)
- Any CICS hexadecimal address (field ❷)
- Single table entry (field ❸)
- Program load module or BMS map (fields ❹ and ❺)

Follow these entry instructions:

- Specify the area to display using **one** of the following entries:

❶ **CICS AREA:**

To get an entire area or table, specify any CORE key area, such as CSA.

❷ **ADDRESS:**

To get a CICS address, specify one to eight hexadecimal digits.

❸ **CICS TABLE entry:**

To get a single table entry, specify the entry next to the appropriate table.

Note: To display the entire table, use the CICS AREA field.

❹ **PROGRAM:**

To get a program load module, specify a program name.

❺ **PROGRAM and M:**

To display a BMS MAP, specify the map name in the PROGRAM field, and specify M in the field to the right.

- After specifying the system areas for display, specify where to begin the display using the Optional offset and/or Scan fields.

❻ **Optional offset:**

Specify a hexadecimal offset or sequence of offsets.
Each offset must begin with either + (a positive hexadecimal displacement), - (a negative hexadecimal displacement), @ (an indirect address), or % (an indirect XA address). You can also calculate offsets. For example: @32*4.

❼/❽Scan Value and Range:

CA InterTest for CICS searches for the value you specify in the range of hexadecimal or decimal bytes you specify.

- To get the requested display, press **Enter** for a structured display. (Structured displays give field names and main storage contents in DSECT sequence.)

  From the structured display, press **Enter** for a dump display. (Dump displays give main storage contents with 16 hexadecimal and character bytes per line.)

Load or Delete Program Modules in Main Storage

To load a program into main storage, follow these steps.

1. Enter the program name in field ❹, **L** in field ❺.
2. Press **Enter** for a structured display.
3. Press **Enter** a second time for a dump display.

To display the program from other than the beginning after it is loaded, follow these steps.

1. Specify an Optional Offset in field ❻.
2. Press Enter for a structured display.
3. Press Enter a second time for a dump display.

To delete a program from main storage, follow these steps.

1. Specify the program name in field ❹, and D in field ❺.
2. Press Enter to display the PPT for the program entry to be deleted.
3. Follow the instructions to confirm the delete.

Change and Search for Data

The change and search functions operate the same as on the Breakpoint-Related Areas menu. For example, the entries in the following screen move a value of hexadecimal 600 into the field located at an offset of (hexadecimal) 12 for the file PROTSYM.

```
CA InterTest CORE COMMAND BUILDER - SYSTEM-RELATED AREAS  (CORE=Syst)

 Specify AREA, ADDRESS, TABLE or PROGRAM to be displayed or changed:

   CICS AREA:  ________   CSA, TCT, SIT                 ADDRESS: ________
                         CWA, DLP, OPFL, ABND

   CICS TABLE entry:    PROTSYM_  TCT ____

   To display PROGRAM:  ________   ENTER L to load, D to delete or M for map: _

 Optional offset: +12_____________________________________

 Scan value: ____________________________________________          Data  Formats
 Scan range: ______            B to scan backwards: _       -----------------
                                                            | P' ' = Packed |
 To VERIFY and/or CHANGE data:                              | X' ' = Hex    |
 Existing data: _____________________________________       | C' ' = Char   |
      New data: X'0600'_____________________________        -----------------


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

The CORE Command Builder Task-Related Areas menu is used to view or modify areas relating to a particular task. To view the menu, select Option **2 Task areas** from the Main Storage menu. The CORE Command Builder Task-Related Areas menu is shown next. Use this menu to display, change, or scan storage relating to a specified task for a task waiting at a breakpoint.

```
CA InterTest CORE COMMAND BUILDER  TASKRELATED AREAS  (CORE=Task)

Specify keyword of AREA to be displayed or changed.

 Task AREA:   ________   DWE, JCA, LLA, STCA, TACB, TCA, TWA, USER, XMTXN, SMX
                        CLOT, RDIIN, SQLCA, SQLRCODE, PLCB, PTA, TASKENT
 Task number: 00052

Optional offset: ____________________________________

Scan value: ____________________________________          Data  Formats
Scan range: ______             B to scan backwards: _
                                                        | P' ' = Packed |
To VERIFY and/or CHANGE data:                           | X' ' = Hex    |
Existing data: ____________________________________     | C' ' = Char   |
     New data: ____________________________________



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

CORE Commands and Advanced Options

Once you become proficient with the CORE command builder menu screens, enter CORE commands directly. The commands are easy to learn after using the menus. All CORE commands created from the menus are presented on CORE displays as a self-teaching aid.

Experienced users often find that entering commands is faster than using the menu screens. Enter **CORE** commands on a clear screen, or on the command line of a Detailed or Source Listing Breakpoint display, or on any CORE display.

The CORE commands are explained in the following sections with some advanced options that are intended only for experienced CA InterTest for CICS users. If you are an inexperienced CORE user, skim the rest of this chapter and reread it when you are more familiar with CORE. Find details on using CORE commands in online help.

Display Main Storage

To display storage starting at a specific location, enter the command:

CORE=location

Specify the location as one of the following names:

- Symbolic name such as a COBOL paragraph name, COBOL data name, COBOL or PL/I statement number, or Assembler label that is defined in a program currently at a breakpoint
- CORE keyword (such as CMAR for the COMMAREA)
- Hexadecimal address

To display storage beginning at a displacement from a particular location, add or subtract the displacement using either of these commands:

```
CORE=location+displacement
CORE=location-displacement
```

To display main storage beginning at a displacement from the area currently displayed, use either of the following commands:

```
CORE+displacement
CORE-displacement
```

displacement

Indicates offset of the first byte of the area to be viewed from the first byte of the area currently displayed. The displacement number is hexadecimal, such as, 100, or decimal, such as, 256T. A number followed by the letter T, for base ten, specifies a decimal number.

Examples

The next examples show a number of CORE commands that are described in following sections:

- To display the COBOL Working-storage area, enter:

  CORE=CWK

- To display the data item EMPLOYEE-NUMBER enter the command shown next. The data item is a variable or fixed length.

  CORE='EMPLOYEE-NUMBER'

- To display main storage beginning hexadecimal 10 bytes (or 16 bytes) after the data location TABLE, enter:

  CORE='TABLE'+10

- To display main storage hexadecimal 100 bytes (or 256 bytes) after the start of the area currently on display, enter:

  CORE+100 or CORE+256T

- To display the storage area that is hexadecimal 100 bytes (or 256 bytes) before the area currently on display, enter:

  CORE-100 or CORE-256T

- To display (for Assembler programs) the area at displacement TIOATDL from the address in the register equated to TIOABAR, enter:

  CORE='TIOABAR.TIOATDL'

- To display the CUSTOMER-NUMBER as found in CUSTOMER-FILE-A, enter:

  CORE='CUSTOMER-NUMBER OF CUSTOMER-FILE-A'

- To display the ZIP-CODE in a specific CITY for a specific CUSTOMER enter the command shown next.. Notice that each comma must be followed by a space.

  CORE='ZIP-CODE(NAME-IDX, CITY-IDX, ZIP-IDX)'

The following table summarizes how you enter locations and displacements in CORE commands. For details,CORE Commands for PL/1 Symbolic Programs (see page 406) later in this chapter.

	COBOL	PL/I	Assembler
Symbolic Location	'data name' 'paragraph name' #statement number CORE keyword	'symbolname‘ # statement number CORE keyword	'regname.label‘ data location 'label‘ – program location or register CORE keyword
Symbolic Displacement	'paragraph name' #statement number	'label‘ # statement number	'label‘

Any Language	
Numeric Location	1- to 8-hexadecimal digits 1- to 8-decimal digits followed by the letter T
Numeric Displacement	1- to 6-hexadecimal digits 1- to 8-decimal digits followed by the letter T (number1*number2) or (number1/number2)
CICS Areas	CORE keyword or keyword. fieldname

Elements of a CORE Command

An element of a CORE command is the minimum amount of symbols and text sufficient to instruct CA InterTest for CICS to perform a specific CORE function. Each CORE command must contain one complete element, although it can contain more than one. A CORE command element begins with any one of the following symbols:

- **Equal Sign (=)** indicates that an absolute address (entered in hexadecimal), a symbolic address, or a CORE command function follows.
- **Plus Sign (+)** adds the number of bytes specified after it to the address in the storage pointer. This advances the storage pointer forward.
- **Minus Sign (-)** subtracts the number of bytes specified after it from the address in the storage pointer. This moves the storage pointer backward.

  Note: The plus and minus signs must be followed by a displacement, specified symbolically, or as a number, a multiplication, or a division.
- **At Sign (@)** works like the plus sign. However, after the address is advanced, it obtains the next address from the three right-most bytes of the four bytes found at the current address. The result is indirect addressing. The at sign uses the lower 24 bits for addressing.
- **Percent Sign (%)** works like the at sign, except it uses 31 bit (XA) addressing. It obtains the address from the four bytes of the fullword.

  Note: The at and percent signs must be followed by a displacement specified symbolically, or as a number, a multiplication, a division, or an absolute or symbolic address.
- **Comma (,)** indicates that the following text is a further description of the previous element and not the beginning of a new element.

There are many elements available to you that are described later in this chapter. In addition, create new elements that apply only to your site.

Note: The *Installation Guide* provides instructions for defining new elements by coding CORUCOM macros.

Storage Pointer

The CORE transaction maintains a storage pointer that is adjusted as CA InterTest for CICS reads and interprets the CORE commands you enter. Before the first input command is processed, this pointer is set to the address of the CICS Common System Area (CSA).

After you enter a command and the first CORE display opens on your terminal, the storage pointer is set to the address of the first displayed byte. This is the address of the storage pointer that the CORE transaction recognizes when it examines the next command you enter. For example, the following sequence of commands locates the area named (symbolically) ACCOUNT-NUMBER in the COBOL program whose task is presently waiting at breakpoint, and displays it. Then it adds five to the address of the displayed area and produces another display that begins at the new address.

```
CORE='ACCOUNT-NUMBER'
CORE+5
```

CA InterTest for CICS adjusts the storage pointer for every element found in your commands. CA InterTest for CICS reads each command from left to right and executes the first executable element it finds.

Example

The following sequence of commands:

```
CORE='CHARGE-ITEMS'
CORE=SCAN404T,C'FRONT PANEL'
CORE=CHGC'REAR PANEL'
```

accomplishes the same thing as the following single command:

```
CORE='CHARGE-ITEMS'=SCAN404T,C'FRONT PANEL'=CHGC'REAR PANEL'
```

Both commands find the field named CHARGE-ITEMS in the program waiting at breakpoint, scan that storage area (which is 404 bytes long) for the characters FRONT PANEL, and change these bytes to contain the characters REAR PANEL instead. (This is an example of using an equal sign to indicate the beginning of a new element.)

The CORE transaction protects itself against incorrect addresses and recovers from abends caused by store- or fetch-protected addresses. It usually encounters these kinds of addresses during the search for the specified storage area, or if you have made a mistake in formulating the CORE command. If this type of error occurs, the message STORAGE ACCESS ERROR displays.

=WHERE directive

The =WHERE directive is optionally specified after the storage pointer. When this directive is specified, the CORE display identifies what CICS subpool, control block, or program owns the storage.

Example:

```
CORE=1234567=WHERE
```

CORE Screen Scrolling Commands

PF7 and PF8 let you scroll backward and forward one page at a time. However, it is possible to scroll forward or backward several pages at a time.

To scroll forward the specified number (1 to 999) of pages, enter:

```
CORE=FWDnnn
```

To scroll backward the specified number (1 to 999) of pages, enter:

```
CORE=BWDnnn
```

Example

The command CORE=FWD5 causes CORE to display the storage area five pages ahead of the current area on display. Each page is the size of one CORE display, which is hexadecimal 110 or decimal 272 bytes. So, the command CORE=FWD5 scrolls the display forward 5 x 272 = 1360 bytes.

Indirect Addressing

The at sign (@) is used to signify indirect addressing in a CORE command. Indirect addressing uses an address stored in a **pointer** field to locate a field for display. The basic command formats used to perform indirect addressing follow.

To display main storage beginning with the address stored at the specified location, enter:

```
CORE=location@0
```

To display main storage beginning with the address stored in the fullword located at the given displacement from the specified location, enter:

```
CORE=loc@disp
```

Note: A displacement following the at sign must be zero or positive. Negative displacements are not allowed.

Examples

The following examples show how to use the at sign.

- Suppose INPUT-RECD-ADDR is the name of the BLL cell used to contain the address of an input record. To display the input record when the COBOL program stops at a breakpoint, enter:

  ```
  CORE='INPUT-RECD-ADDR'@0
  ```

This command has the same effect as:

CORE='INPUT-RECORD'

INPUT-RECORD

Indicates the data structure addressed by the BLL cell named INPUT-RECD-ADDR.

- To display the first storage area on the task storage chain, enter:

CORE=STCA@4

STCA

Indicates CORE keyword for the System Task Control Area. CORE displays the storage area whose address is stored in the low-order three bytes of the fullword at STCA+4. The result is that CORE displays the first storage area on the task storage chain, which is stored at STCA+4. Pressing Enter repeats the CORE command, letting you run through the entire chain.

View Storage for a COBOL or PL/I Program at a Breakpoint

Use names defined in a COBOL or PL/I program in CORE commands as follows:

- A COBOL data name, paragraph name, or PL/I symbol name must be in single quotes; qualify, index, or subscript a COBOL data name
- A number sign (#) must directly precede a statement number

Notes:

- If the same data name, such as CUSNAME, is defined more than once in a COBOL or PL/I program, reference particular occurrences using the qualified data name.
- CORE displays as many bytes as are defined by a symbolic name unless the name is modified by a displacement.

Example

To get a full CORE display (272 bytes of storage) beginning with the data location called CUSNAME, enter the command CORE='CUSNAME'+0, not just CORE='CUSNAME'.

- If you have a data name defined in arrays, specify it like this:

```
CORE='NAME-1(X1, X2, X3)'
```

Examples

In each of the following examples, the symbolic program addresses relate to the program currently stopped at a breakpoint. Using symbolic addresses for a program not stopped at a breakpoint is described in Display Program Code for a Program Not at a Breakpoint (see page 419) later in this chapter.

- To display the data field CUSTNAME in the program (COBOL) that is currently stopped at a breakpoint, enter:

  ```
  CORE='CUSTNAME'
  ```

- If you press **PF12**, the COBOL structure in a DMAP-defined sequence is displayed. A COBOL structure is defined by level numbers.

Example

A structure is finished when a data item's level number is equal to or less than the level number of the first item in the structure.

- To display the first executable instruction in the named COBOL paragraph of the program currently stopped at a breakpoint, enter:

  ```
  CORE='900-END-RTN'
  ```

- To display the data field TASK_TEXT in the program (PL/I) that is currently stopped at a breakpoint, enter:

  ```
  CORE='TASK_TEXT'
  ```

- To display the code for the first executable instruction in statement 313 of the program currently stopped at a breakpoint, enter:

  ```
  CORE=#313
  ```

- To display the code for the second verb in statement 313 of the COBOL program currently stopped at a breakpoint, enter:

  ```
  CORE=#313.1
  ```

  Note: The first verb is entered as CORE=#313.0.

- To display the code for the first executable instruction in the program currently stopped at a breakpoint, enter:

  ```
  CORE=#1
  ```

- To display the code for a qualified program name currently stopped at a breakpoint, enter:

  ```
  CORE='ACME300N:WS-VARIABLE-FLD'
  ```

Display Qualified, Variable Length, Indexed, or Subscripted COBOL Data Names

Specify qualified, variable length, indexed, and subscripted COBOL data names as follows:

- To display non-uniquely defined COBOL data names, enter:

  ```
  CORE='NAME-1 OF NAME-2'
  ```

- Specify up to seven levels of indexing or subscripting.
  - To display a variable length COBOL data item, such as a data item declared with an 'OCCURS X to Y times depending on Z' clause, enter:

    ```
    CORE='DATA-NAME'
    ```

  - To display an indexed or subscripted COBOL data item defined in arrays, enter:

    ```
    CORE='NAME(X1, X2, X3)'
    ```

- Follow these syntax rules:
- The data name must be enclosed in quotes.
- Specify up to seven levels of indexing or subscripting for a data item.
- Parentheses, (...), indicate the beginning and end of indexing and subscripting.
- X1, X2, X3 are index names, subscript names or numeric subscript values. If more than one item is specified, a comma and a blank must precede the next item.
- Do not mix indexing and subscripting in the same command.

Example

To display a specific entry within a table, enter:

```
CORE='TABLE(1, SUB-2)'
```

To display the CUSTOMER-NUMBER as found in CUSTOMER-FILE-C, enter:

```
CORE='CUSTOMER-NUMBER OF CUSTOMER-FILE-C'
```

To display the ZIP-CODE in a specific CITY for a specific CUSTOMER, enter:

```
CORE='ZIP-CODE(NAME-IDX, CITY-IDX, ZIP-IDX)'
```

View Storage for an Assembler Program at a Breakpoint

Use Assembler labels in a CORE command to symbolically address data locations, program locations, or registers as shown in the following table.

Location	Format
data locations	'regname.label'
program locations	'label' (the label value must exceed 15)
registers	'regname' (where the equated value is less than 16)

Examples

Each format is shown in the following examples, where the symbolic program addresses relate to the Assembler program currently stopped at a breakpoint.

- To display the area at a displacement of TIOADBA from the address in the register equated to TIOABAR, enter:

  ```
  CORE='TIOABAR.TIOADBA'
  ```

- Press PF12 to display an Assembler DSECT structure sequence.
- To display the program code starting at the label SCANLOC, enter:

  ```
  CORE='SCANLOC'
  ```

- To display the contents of the register equated to TIOABAR, enter:

  ```
  CORE='TIOABAR'
  ```

Use a Different Module for Assembler Structures

When requesting an assembler structure display, you might encounter a not found condition for one of these reasons:

- The data area's DSECT was not printed in the Assembler output listing
- The requested DSECT is in an area preceded by a PRINT OFF instruction

To handle these occurrences, do the following things:

- Create an Assembler job containing your common DSECT layouts and use the CA InterTest for CICS post processor step
- Create one program that contains all of your DSECTS with the PRINT ON option

The following considerations are important:

- Do not specify a LISTER= option
- You do not have to link edit the module
- You do not need a PPT Entry (in CICS) for this module
- The name of this module must be unique, and it must not be the same as any other module in the Symbolic File

Reference this module online by adding USE=xxxxxxxx to the CORE command. This additional option tells CA InterTest for CICS to access the named module (xxxxxxxx) instead of the program where CA InterTest for CICS is currently stopped.

- The data area's DSECT was not printed in the Assembler output listing
- The requested DSECT is in an area preceded by a PRINT OFF instruction

To handle these occurrences, do the following things:

- Create an Assembler job containing your common DSECT layouts and use the CA InterTest for CICS post processor step
- Create one program that contains all of your DSECTS with the PRINT ON option

The following considerations are important:

- Do not specify a LISTER= option
- You do not have to link edit the module
- You do not need a PPT Entry (in CICS) for this module
- The name of this module must be unique, and it must not be the same as any other module in the Symbolic File

Reference this module online by adding USE=xxxxxxxx to the CORE command. This additional option tells CA InterTest for CICS to access the named module (xxxxxxxx) instead of the program where CA InterTest for CICS is currently stopped.

Example

The command CORE='R12.DFHTCADS',USE=TESTMAST displays the structure for the USER TCA with the DFHTCAD DSECT. The address of the USER TCA is contained in register R12 with its DSECT structure saved in the Symbolic File records named TESTMAST. If R12 is not defined in your program, CA InterTest for CICS assumes you meant register 12. (The same assumption would be made for R0 through R15.) If the field name, such as DFHTCADS, has not been defined, CA InterTest for CICS searches for the default CICS structure for the field.

CORE Commands For PL/I Symbolic Programs

If you have purchased the PL/I Symbolic Option and are currently stopped at a breakpoint, use CORE to view storage associated with a symbolic PL/I program. The PL/I Symbolic CORE facilities work just like the COBOL Symbolic facilities of CORE, with the following differences:

- There are three PL/I symbolic formats for CORE commands:

  ```
  CORE='symbolname'
  ```

  This command checks to see if the symbol is a label. If it is, the machine code at that location is displayed. If it is not, the definition of the symbol in the most recently activated procedure or block is used to display the current value of the symbol.

  Note: The most recently activated block, containing a definition of the symbol, need not be the currently executing block.

  ```
  CORE='blockname:symbolname'
  ```

  ```
  CORE='blockname:name1.name2.name3'
  ```

  This command checks to see if the symbol is a label in the qualifying block. If the named block is active, the definition of the symbol in that block is used to display the current value of the symbol. If it is not, the blockname is checked to verify that it is on the active block chain.

  ```
  CORE='symbolname1 >symbolname2'
  ```

  This command displays variables listed as based (*) in the PL/I Cross Reference, where:

  - symbolname1 is assumed to be a pointer that points to an area to be mapped by symbolname2. If it is not a pointer, then the address of symbolname1 is mapped by symbolname2.
  - symbolname2 formats the data to be displayed. In the case of a structure, symbolname2 is assumed to be the beginning of the area pointed to by symbolname1.

    ```
    CORE='name1.name2(name3,name4)'
    ```

  Display array item indicated by the resolution of variables name1 and name2. Up to 15 levels of qualification and subscripting are supported.

- Controlled variables (CTL) are not supported.
- Symbolic CORE commands should not be used when a program is stopped at a PROC or BEGIN statement. The DSA environment is not properly established by PL/I until the first executable statement in a block is actually executed. As a result, CORE provides unpredictable results if you refer to a symbol that has not been allocated or freed (such as BASED variables that are allocated in the block).

- Like named variables are supported as follows:
  - If two variables with the same name are declared in the same block/procedure, attempting to display these variables with a CORE command requires qualification.

    ```
    CORE='name1.name2'
    ```

  - If two variables with the same name are declared in different blocks/procedures, display them using the format:

    ```
    CORE='blockname:symbolname'
    ```

- Based variables are supported with an unlimited number of levels of basing.

Example

See this sample in the following examples.

```
 PROC1:  ...;
  DCL 1    A,
           2 FIELDB  ...,
             3 FIELDC ...;
SUBPROC:  ...;
  DCL 1   A,
          2 FIELDB  ...,
            3 FIELDC ...;
END SUBPROC;
```

To display the variable FIELDB declared in PROC1, enter:

```
CORE='PROC1:FIELDB'
```

To display the variable FIELDB declared in SUBPROC, enter:

```
CORE='SUBPROC:FIELDB'
```

See the following sample for the next example.

```
DCL   Q PTR;
DCL   R PTR;
DCL   Z BASED(Q);
```

To display Z using R as the pointer, code the CORE command as follows:

```
CORE='R >Z'
```

Display Program Code for a Program Not at a Breakpoint

When a symbolic name is used in a CORE command, CORE must know in which program the name or label is defined. If the program is currently stopped at a breakpoint, CORE assumes that program will be displayed; otherwise, the following command format must be used:

```
CORE=PGM=progname+symbolic name
```

Example

To display the code generated in the COBOL paragraph named EDIT DATA in the resident program named ORDENTRY, which is not at a breakpoint, enter:

```
CORE=PGM=ORDENTRY+'EDIT DATA'
```

This command adds the displacement of EDIT DATA to the address of the first byte of the program ORDENTRY and displays main storage starting at that address. If the program is not resident, specify the following command to load the program into storage:

```
CORE=LOAD=progname
```

Breakpoint Commands

Certain CORE commands are applicable only when a task is stopped at a breakpoint. These commands let you do the following tasks:

- Display storage areas whose contents are meaningful only when task execution halts at a breakpoint
- Display the screen as it existed before the breakpoint or restore the breakpoint display
- Dynamically acquire main storage
- Change the values of program data items using COBOL like MOVE commands

Display Storage Related to a Task at a Breakpoint

The following table lists the commands used to display an area of main storage whose contents are meaningful only when a task stops at a breakpoint.

Core Command	Result
CORE=ARGnn	Displays argument nn on the breakpointed program's argument list, where nn is a decimal number from 0 99. The address of the argument list must be in register 1.
CORE=BLLnn	Displays the BLL cell with relative number nn, where nn is a decimal number from 0 99. The first BLL cell is specified in the CORE command as BLL1, the next as BLL2, and so on.
CORE=BLLS	Displays the COBOL program's BLL cells.
CORE=BLXnn	Displays the BLX (external program storage) cell with relative number nn, where nn is a decimal number from 0 99. The first BLX cell is specified in the CORE command as BLX0, the next as BLL1, and so on.
CORE=BLXS	Displays the COBOL program's BLX (external program storage) cell pointers.
CORE=CLOT	Displays the CICS Life Of Task block. This area exists only after the first SQL CALL in your program (z/OS only).
CORE=CMAR	Displays the CICS EXEC Communication Area (DFHCOMMAREA) owned by the breakpointed command level CICS task.
CORE=CURR	Displays the current instruction about to be executed when the task was halted at a breakpoint.
CORE=CWK	Displays the breakpointed COBOL program's WORKING STORAGE.
CORE=DSA	Displays the breakpointed program’s Dynamic Storage Area.
CORE=EIB	Displays the EXEC Interface Block owned by the breakpointed command level task. **Note:** When using the Integrated CICS translator of Enterprise COBOL for z/OS, this command will not work unless the NOLINKAGE translator option or the ADATA compiler option is specified.

Core Command	Result
CORE=EIS	Displays the EXEC Interface Structure owned by the breakpointed command level task.
CORE=FCAR	Displays the Facility Control Area. This is the TCT entry for a terminal oriented task or the DCT entry for a task initiated by a transient data trigger level.
CORE=FLPT	Displays the four doublewords where floating point registers 0 6 were saved. Use this keyword only if the save floating point registers facility was generated for CA InterTest for CICS.
CORE=ITBE	Displays the instruction that was to be executed when the task halted at the breakpoint.
CORE=LCL	Displays the breakpointed COBOL program's LOCAL-STORAGE.
CORE=MXR	Displays the current value of the counter for CICS requests.
CORE=MXS	Displays the current value of the counter for the amount of CICS storage acquired by the program.
CORE=PREV	Displays the last executed Assembler instruction.
CORE=PREVCORE=RDIIN	Displays the Application Invocation Parameter List. The task must have issued an SQL CALL for CORE=RDIIN to function (z/OS only).
CORE=REGS	Displays the 16 fullwords where registers 0 15 are saved.
CORE=Rnn	Displays the contents of register nn prior to the breakpoint.
CORE=SQLCA	Displays the SQL Communications Area. This area contains data only after the first SQL call in your program (z/OS only).
CORE=SQLRCODE	For z/OS only: Displays the contents of SQLCODE as a decimal number and converts this value into its associated text message. Explanations of any SQL warning codes are also displayed. For a DSNC abend, the failure code, the failing DB2 subcomponent identifier, and the subsystem termination reason code are displayed with explanations of the failure code and the subsystem identifier.

Core Command	Result
CORE=SSCR	Displays the last CICS screen displayed by your program before it stopped at the current breakpoint.
CORE=TAL	Displays the current value of the TALLY counter.
CORE=TGT	Displays the COBOL program's Task Global Table.
CORE=TIOA	Displays the first Terminal I/O Area chained to the breakpointed task.
CORE=TUAR	Displays the saved user area in the terminal table (TCT) entry of the breakpointed task. Use this keyword only if STUAR=YES was specified when CA InterTest for CICS was installed.
CORE=WKAR	Displays the CA InterTest for CICS Work Area of the breakpointed task.

Display Storage for an Inactive COBOL Program at a Breakpoint

The following table lists the command that can be used to display areas of main storage for both **inactive** and **active** COBOL programs when at a breakpoint.

Core Command	Result
CORE=BLLnn=program	Displays the BLL cell for the program specified on the command, where nn is a decimal number from 1 99. The first BLL cell is specified in the CORE command as BLL1, the next as BLL2, and so on.
CORE=BLLS=program	Displays the COBOL program's BLL cell storage pointers.
CORE=BLXnn=program	Displays the BLX (external program storage) cell for the program specified on the command, where nn is a decimal number from 0 99. The first BLX cell is specified in the CORE command as BLX0, the next as BLX1, and so on.
CORE=BLXS=program	Displays the BLX cells for the external storage for the program specified on the command.
CORE=CWK=program	Displays the COBOL working storage for the program specified on the command.
CORE=TGT=program	Displays the Task Global Table for the program specified on the command.

Restore the Screen at a Breakpoint

To display the screen image as it existed prior to the breakpoint, enter:

```
CORE=SSCR
```

To return to the CORE command after displaying the screen image, press **Clear**.

Dynamically Acquire Main Storage at a Breakpoint

To dynamically acquire main storage when a program is at a breakpoint, enter:

```
CORE=GETM,L=len,CL=type,INIT=xx
```

L=len is mandatory. The value of len is the same as the one entered with the parameter LENGTH(...) in a CICS GETMAIN command or would be entered with the keyword NUMBYTE=... of a GETMAIN macro. It could be a hexadecimal value or a decimal followed by the letter T.

CL=type is optional and specifies the type of storage to be acquired. CL=USER acquires user class storage and CL=TERM acquires terminal class storage (TIOA). If this parameter is omitted, CL=USER is assumed.

INIT=xx is optional. The hexadecimal digits xx specify the character used to initialize the storage area. The default is binary zeroes.

This command is equivalent to an EXEC CICS GETMAIN or a DFHSC TYPE=GETMAIN macro.

When a CORE=GETM command executes, CA InterTest for CICS displays the acquired area beginning with its eight byte Storage Accounting Area (SAA). The address of the acquired area must be saved in a register where the program expects to find it.

The SET command is useful to load the address. Note that if the address is placed in a COBOL BLL cell, the address must be adjusted up 8 bytes to point past the SAA. If a COBOL program has loaded the contents of the BLL cell into a register, the register must also be loaded.

Examples

The SET command is expressed as follows:

```
CORE+8=SET=BLL1
CORE=SET=REG6
CORE=SET='SSA POINTER'
```

COBOL Like MOVE Command at a Breakpoint

The MOVE command is valid for COBOL and PL/I programs stopped at a breakpoint.It has two formats:

- To move the contents of field 1 to field 2, enter:

  ```
  CORE=MOVE field 1 TO field 2
  ```

- To move a literal to a field, enter:

  ```
  CORE=MOVE literal TO field 2
  ```

field 1

Names the sending area.

field 2

Names the receiving area. Both fields must be defined in the program. Do not enclose either field name in quotes. Qualify, index, or subscript either field.

literal

Indicates a fixed point numeric value, a string of EBCDIC characters, or a keyword, such as ZERO or SPACES.

fixed point literal

Indicates a string of up to 18 decimal digits with an optional leading plus, +, or minus sign, . (No sign defaults to +.) Embedded decimal points are not allowed.

Specify any string of characters from one to 120 EBCDIC characters enclosed in quotes as a literal. A quotation mark cannot be part of the character string defining the literal.

Use the following keywords as literals:

Literal Keyword	Actual value
ZERO	Numeric value 0
SPACE	One or more blanks (X'40')
HIGH-VALUE	One or more occurrences of X'FF'
LOW-VALUE	One or more occurrences of X'00'
QUOTE	One or more occurrences of the quotation mark (X'7D')

Note: Append the letter S to any of the previous keywords without changing its meaning. For example, the command MOVE SPACES TO CUSTNAME is equivalent to MOVE SPACE TO CUSTNAME.

To repetitively fill a field with a particular character string or decimal digit, precede the literal with the keyword ALL followed by a space. For example, ALL 'XYZ' or ALL 9. The literal following ALL is any valid EBCDIC character string enclosed in quotes. If the literal following ALL is a numeric, it must be a single unsigned decimal digit.

The MOVE command handles only certain data types and makes any required conversions from one data type to another. The conversions follow the examples specified in *the IBM Full American National Standard COBOL Reference Manual*. The fields are checked for numeric content, as needed.

A list of supported data types and the corresponding conversion rules follows:

Data Type	Conversion Value
GROUP	Data is left justified with no conversion. This type of move is made if either the sending or receiving field has group data.
DISPLAY	Data conversion is performed with left justification.
DISPLAY NUMERIC	Data conversion is performed with right justification.
COMP-3	Data conversion is performed.
COMP	Data conversion is performed.
ALPHANUMERIC EDITED	Handled as DISPLAY, but without editing.
NUMERIC EDITED	Handled as NUMERIC, but without editing.

Display Task Owned Areas

To display areas of storage that are related to a particular task, use the keywords in the following table. Unless you specify otherwise, the areas displayed relate to the task that created the most recent breakpoint display.

Use the command form:

```
CORE=Area Displayed
```

Area Displayed is one of the following CORE keywords:

CORE Keyword	Task-Related Storage Area
DWE	Top Deferred Work Element
JCA	Top Journal Control Area
LLA	Top Load List Area
PLCB	Program Logical Level Control Block
PTA	Program Transaction Area Control Block
SMX	Storage Manager Transaction Control Block
STCA	System portion of TCA
TACB	Top Task Abend Control Block
TASKENT	Kernel Domain Task Definition Control Block
TCA	User portion of TCA
TWA	TWA portion of TCA
USER	Top USER class storage area
XMTXN	Transaction Manager Transaction Definition Control Block

Set the Task Number in a CORE Command

The commands listed in the previous table display the task area relating to the task that most recently sent a breakpoint display to the terminal screen. To display storage for a different task, you must enter a task number in the CORE command. The new task number remains in effect until you change it or a new breakpoint display is received at the terminal. The task number is in the upper right corner of CORE displays.

To specify the task number to be used by CORE in accessing task related areas, enter:

```
CORE(task no)
```

To reset the task ID to the task currently waiting at a breakpoint, enter:

```
CORE(*)
```

Use this command if there is just one task currently at a breakpoint at the terminal.

Examples

These examples show how to display storage for non breakpointed tasks.

- To display the top USER-class storage area for task 2145, enter:

  ```
  CORE(2145)=USER
  ```

  Subsequent task related commands refer to task 2145 until the task number is changed in a CORE command or until a new breakpoint occurs.

- To display the user portion of the TCA for the task currently at a breakpoint, enter:

  ```
  CORE(*)=TCA
  ```

Display Any CICS Control Area Structure in the Symbolic File

Use the following command to display any CICS control area in the Symbolic File, and to display the structure of the first entry in a control area:

```
CORE='CICS CONTROL AREA NAME'
```

Examples

These examples demonstrate how to display entries located on the CSA.

- To display the structure at the first entry in a control area, enter:

  ```
  CORE=CSA
  ```

- To display the structure at a specific entry (CSADATFT) in the CSA, enter:

  ```
  CORE=CSA.CSADATFT
  ```

Display System-Related Areas

The following CORE keywords display areas of main storage that contain system related information:

CORE Keyword	System-Related Storage Area
ABND	Kernel Error Data Block (CICS 3.2 and higher only)
CSA	Common System Area
CWA	Common Work Area
DLP	DL/I Interface Parameter List
FCT	First entry of the File Control Table
FILE=filename	FCT entry for the file specified by filename
MAP=mapname	Displays a BMS Map
MAP=name(mapset)	Displays a BMS Map belonging to the mapset
OPFL	Optional Features List of CICS
PGM=progname	Area occupied by the named program
SIT	CICS System Initialization Table
TCT	Terminal Control Table address list
TERM=termid	TCT entry for the terminal specified by termid
TERM=*	TCT entry of the terminal where CORE is currently executing

Change the Contents of Main Storage

Use the following commands to change the contents of main storage. However, use CORE menus to accomplish the same functions.

Commands that move data into a specified storage area are listed in the following table:

Command	Function
MOVE	Used when a COBOL or PL/I program is stopped at a breakpoint. This command was described earlier in COBOL Like MOVE Command at a Breakpoint.
CHG	Used to specify that data be moved into main storage at the current or specified main storage location.

Command	Function
MOVEIN	Used to move data from one location in storage to another.
SET	Used to load an address into the low order three bytes of a specified fullword.

The CHANGE Command (CHG)

The CHG command is used to move new data into main storage. The new data is specified in the command after the keyword CHG in character, hexadecimal, packed, or binary format. This command has two formats:

- To move a data string into the main storage area that begins with the first byte of the current CORE display, enter:

  ```
  CORE=CHGdata
  ```

- To move data into main storage beginning at the specified location, enter:

  ```
  CORE=loc=CHGdata
  ```

Specify the data in a CHG command using any of the following formats:

- C'chardata' for character data
- X'hexdata' for hexadecimal data
- P'value' for packed decimal data
- F'value' for a binary fullword
- H'value' for a binary halfword preceded by a plus or minus sign

Replace 'value' with a decimal number, optionally preceded by a plus (+) or minus ( ) sign.

Examples

To place 123 in the first three bytes of storage on display, enter:

```
CORE=CHGC'123'
```

To perform the same function as in Example 1, enter:

```
CORE=CHGX'F1F2F3'
```

To place the characters 123 in the first three bytes of the data field ACCT NUM defined in the program currently stopped at a breakpoint, enter:

```
CORE='ACCT NUM'=CHGC'123'
```

To move the two byte packed decimal value of 100 into the first two bytes of the data field QUANTITY, enter:

```
CORE='QUANTITY'=CHGP'100'
```

When entering an apostrophe as part of character data, the apostrophe must be repeated.

To move the name O'HARA into the data area CUSNAME, specify:

```
CORE='CUSNAME'=CHGC'O"HARA'
```

Enter any combination of hexadecimal and character data.

To change the value of RECKEY to the name SMITH followed by two bytes of binary zeroes, specify:

```
CORE='RECKEY'=CHGC'SMITH'X'0000'
```

Bit Manipulation

Appending OC, NC, or XC after the data string lets you perform one of the following logical operations to the contents of main storage:

- OC to OR the data in the command to storage
- NC to AND the data in the command to storage
- XC to EXCLUSIVE OR the data in the command to storage

Examples

The following examples demonstrate how to manipulate main storage bits.

- To set the high order bit of the first byte on the current CORE display to 1, enter:

  ```
  CORE=CHGX'80'OC
  ```

- To set the high order bit of the first byte on the current CORE display to 0, enter:

  ```
  CORE=CHGX'7F'NC
  ```

- To reverse all the bits of the first byte on the current CORE display, enter:

  ```
  CORE=CHGX'FF'XC
  ```

Verify Storage Before a Change

Use the VER command with a CHG command to prevent an incorrect change to storage. This command verifies that the specified data is stored at the specified location. Specify the VER command as follows:

```
CORE=loc=VERdata
```

The data in a VER command is specified using the following formats:

C'chardata'— character data

X'hexdata'—hexadecimal data

Example

Verify that the NAME field contains JIM before changing it to JAY. If JIM is not in the NAME field, the change command is rejected, enter:

```
CORE='NAME'=VERC'JIM'=CHGC'JAY'
```

Move Data with the MOVEIN Command

The MOVEIN command, which lets you move data, has two formats:

- To move data from the FROM location to the area currently displayed, enter:

  ```
  CORE=MOVEIN,L=len,FROM=location
  ```

- To move data from loc2 to loc1, enter:

  ```
  CORE=loc1=MOVEIN,L=len,FROM=loc2
  ```

The length parameter, L=len, specifies the number of bytes to be moved. The maximum value for len is decimal 256, specified as 256T.

Example

To move twenty (hexadecimal 14) bytes of data from NEWNAME into CUSTNUM, enter:

```
CORE='CUSTNUM'=MOVEIN,L=14,FROM='NEWNAME'
```

Scroll an Address with the SET Command

Specify the SET command to set an address as follows:

```
CORE=addr1=SET=addr2
```

The address specified by addr1 is placed into the three low order bytes of the fullword whose address is given by addr2. The high order (left most) byte is filled with binary zeroes.

Examples

These examples demonstrate how to use the SET command.

- To fill the fullword at F12DE8 with the data 00234ABC, enter:

  ```
  CORE=234ABC=SET=F12DE8
  ```

- To put the starting address of the program named PROGRM1 into the fullword that is at a displacement of 2C from the beginning of the program named PROGRM2, enter:

  ```
  CORE=PGM=PROGRM1=SET=PGM=PROGRM2+2C
  ```

Scan Main Storage

The SCAN and SCAB commands search main storage for specified data. SCAN searches forward through storage, and SCAB searches backward. The command formats let you search from the current CORE display or from a specified location.

- To begin a search at the first byte of the current CORE display, enter:

  ```
  CORE=SCANrange,data
  ```

  or

  ```
  CORE=SCABrange,data
  ```

- To begin a search at the location specified as loc, enter:

  ```
  CORE=loc=SCANrange,data
  ```

  or

  ```
  CORE=loc=SCABrange,data
  ```

The search terminates when the first occurrence of the specified data is found or the number of bytes specified in the range have been scanned. Specify the range as a hexadecimal number or a decimal number followed by the letter T.

Examples

These examples show how to search for and replace data in main storage.

- To search for the string STEVE beginning at the location TABLE1, enter:

  ```
  CORE='TABLE1'=SCAN1000T,C'STEVE'
  ```

  The search for STEVE proceeds forward for a maximum of (decimal) 1000 bytes. If STEVE is not found a message displays.

- To search for STEVE starting at TABLE1, and to replace STEVE with the character string CHRIS, enter:

  ```
  CORE='TABLE1'=SCAN1000T,C'STEVE'=CHGC'CHRIS'
  ```

  Note: This command only replaces the first occurrence of STEVE with CHRIS.

Load and Delete Programs or Displaying BMS Maps

To load and delete programs and to display BMS maps, use the CORE commands specified following.

- To load the named program or increase its current use count by 1, enter:

  ```
  CORE=LOAD=progname
  ```

- To inform CICS that the named program can be deleted or to decrease its current use count by 1, enter:

  ```
  CORE=DLTE=progname
  ```

- To display the named BMS map, enter:

  ```
  CORE=MAP=progname
  ```

If a program name contains any of the special characters, @, #, or $, you must enter the program name as C'progname'. For example, to load the program named FIND#7 into main storage, enter:

```
CORE=LOAD=C'FIND#7'
```

Dump Main Storage

Use either of the two formats shown following to dump main storage. The dump is in the format of a partial CICS transaction dump, and is written to the CICS dump data set.

- To create a dump of the area currently on display, enter:

  ```
  CORE=DUMP,L=len,ID=code
  ```

- To create a dump of the area beginning at the specified location, enter:

  ```
  CORE=loc=DUMP,L=len,ID=code
  ```

L=len

Specifies the size of the dump. Specify as a hexadecimal number or a decimal number followed by a T. This parameter is optional and, if omitted, defaults to the size of a CORE display.

ID=code

Specifies a dump code. If omitted, the default is CORE.

Following the dump ID, include an optional message in the DUMP command. A blank must separate the message from the command. The entire DUMP command and any messages are included in the dump.

Example

To produce a dump of the storage area that was on the CORE display screen at the time the command was entered, enter:

```
CORE=DUMP,ID=MY01 BAD ACCT NUMBER
```

The dump ID will be MY01, and the message BAD ACCT NUMBER displays in the dump.

Chain CORE Commands

In general, if you need to execute several CORE commands to achieve a desired result, chain the commands together into one command. The format for chaining commands together is:

```
CORE=command1=command2=...=commandn
```

This is equivalent to sequentially executing the commands as follows:

```
CORE=command1
CORE=command2
CORE=commandn
```

Examples

The following examples demonstrate how to join CORE commands:

- To search the third storage area on the task's storage chain, enter:

  ```
  CORE=STCA@4@4@4=SCAN2000,C'RONALD'=MOVEIN,L=6,FROM=256ABC
  ```

  If the character string RONALD is found within hexadecimal 2000 bytes, CA InterTest for CICS replaces it with whatever is at address 256ABC.

- To get a one hundred byte storage area for the task that is waiting at a breakpoint, and set the address of the first byte (after the eight byte long Storage Accounting Area) into the fullword at the address in register 6, enter:

  ```
  CORE=GETM,L=100T+8=SET=R6@0
  ```

The CALC Option

When viewing a CORE display, overtype the TransID CORE with CALC to use the basic hexadecimal calculator function of CA InterTest for CICS. The CALC function displays the hexadecimal and equivalent decimal number for the given value on the bottom line of the screen. The current CORE display is not affected. Return to the CORE transaction by replacing CALC with CORE.

Note: When using the CALC function, the letter T, for base ten, must follow decimal numbers.

Examples

The following examples demonstrate how to display hexadecimal and decimal values of specified numbers.

- To display the hexadecimal value of 200 and the decimal value of 512 on the bottom line of the screen, enter:

  ```
  CALC=200
  ```

- To display the decimal value of 512 and the hexadecimal value of 200 on the bottom line of the screen, enter:

  ```
  CALC=512T
  ```

- To display the hexadecimal value of 202 and the decimal value of 514 on the bottom line of the screen, enter:

  ```
  CALC=0+(10*16T)+(16T*16T)+2
  ```

Convert PL/I Statement Numbers into Displacements

PL/I users who want to set breakpoints using hexadecimal offsets (displacements) need to use an offset calculated from the beginning of the main control section. There are two ways to get such offsets:

- Use the offset given in the Assembler like listing provided by the source listing option of the PL/I compiler. However, this listing is rather long.
- Use the offset listed in the Tables of Offsets and Statement Numbers portion of a PL/I listing for a statement. This offset is relative to the beginning of the machine code for the internal procedure. Then, calculate the offset relative to the main control section using the following CORE command:

  ```
  CORE=PGM=pgmname;+'procname'+offset
  ```

pgmname

Indicates the PPT ID of a PL/I program module. It must be followed by a semicolon and a plus sign.

procname

Indicates the name of an internal PL/I procedure as listed in the Tables of Offsets.

offset

Indicates a hexadecimal offset for a statement as listed in the Tables of Offsets and Statement Numbers.

This command accepts the offset for a statement as listed in the Tables of Offsets, and returns the hexadecimal value of the offset of that statement as counted from the beginning of the main control section. Use the returned value to set a breakpoint or to request a CORE display of the program code at that location.

Examples

An offset must be specified in the command. (Specify 0 as +0.)

```
CORE=PGM=TESTPROG;+'EDIT INPUT'+10
```

The named program must reside in main storage. If it does not, use the following command to load it:

```
CORE=LOAD=pgmname
```

When you are finished, use the following command to remove the program from main storage:

```
CORE=DLTE=pgmname
```

CORE(NNNNN)=BMSG

Use the CORE(NNNNN)=BMSG command to view a breakpoint display that is currently being displayed on another terminal. By replacing NNNNN with the task number of the breakpointed task, you can view the breakpoint display already showing for that task, and you will be able to display and modify storage related to that task as if your terminal was the breakpoint terminal. You will not be able to resume execution or abend the task at the breakpoint (this must be done from the actual breakpoint terminal). Use the CNTL=DISC command from the command line to disconnect from the breakpoint display.

Chapter 11: Accessing Auxiliary Storage: FILE

Option 4 Auxiliary Storage, from the Primary Option Menu, invokes the CA InterTest for CICS FILE transaction. This FILE transaction lets you display, add, change, and delete records in VSAM and BDAM files, DL/I, DB2, and SQL/DS databases, temporary storage, and transient data.

The FILE transaction is very useful when you want to update records or create test data. Use FILE at:

- Any time, even when the rest of CA InterTest for CICS is inactive
- A breakpoint to inspect and modify data values and generate more test data

The changes you make to a file or database are always permanent, rather than dynamic, changes.

FILE Capabilities

Use FILE to perform the following functions. Each main function is an option on the Auxiliary Storage Menu.

Note: The functions you perform depend on your security authorization.

- Files (VSAM and BDAM)
  - View, add, update, and delete records
  - Browse a file
  - Search for data
  - Mass insert records into a VSAM file
- DB2 and SQL/DS Databases
  - View a table
  - Update columns in a table
  - Delete rows from a table
  - Insert a row in a table
  - Issue other SQL commands, including CREATE, ALTER, DROP, COMMENT ON, LABEL ON, GRANT, and REVOKE

- DL/I Databases
  - Retrieve segments
  - Add, update, and delete segments
- Transient Data
  - Retrieve a transient data record
  - Add a transient data record
  - an intrapartition data queue
- Temporary Storage
  - View or add a temporary storage record
  - View or add a temporary storage queue record
  - Search a temporary storage queue for data
  - Purge a temporary storage queue

In addition to the previous listed functions, use FILE to perform tasks that would otherwise require specially written programs.

Example

Use FILE to do the following tasks:

- Copy a record from one file to another
- Create test data from production files

FILE performs these tasks because it retains retrieved record and segment data in its own WORK AREA.

The CA InterTest for CICS FILE transaction is comprehensive. It enables you to perform interactively any function that is supported by CICS File Control and DL/I, and most DB2 functions. FILE uses CICS services exclusively; it does not use the operating system's data access methods directly.

Remote VSAM File and Temporary Storage Support

Use FILE to access and update remote VSAM files and temporary storage. FILE supports the following files and storage:

- All VSAM file types
- Auxiliary and main temporary storage queues

- Temporary storage queue names of all valid lengths. If you are running CTS Version 1.3, queue names are up to 16 bytes or 32-hexadecimal characters long.

No additional user input is required.

Note: Both sites must be at CA InterTest for CICS r4.2 or higher.

FILE Work Area

FILE keeps record or segment data in its own work area when performing an I/O operation. An *input operation*, such as a READ, places the requested data record or segment in FILE's work area and displays it on the screen.

Consider the following operations when working with the FILE work area:

- A successful input operation overwrites data currently in the work area.
- A successful output operation leaves all data in the work area intact. Reuse the contents of the work area for any number of consecutive output operations without having to reacquire the same data.
- Changing data in the work area does not affect the contents of the accessed data structure (file, database, temporary storage, transient data) until a successful output operation is performed.
- The contents of the work area can exceed the display capacity of the terminal screen. Display any portion of the work area by entering the location in the LOC field or using PF keys. For a complete description of the LOC field, see Dump Format (see page 447) later in this chapter.

The FILE Transaction and Menus

The tasks described in this section apply to any type of data organization. Tasks specific to a particular type of data organization are discussed in the appropriate section.

Access File Facilities

Use the following methods to access the File facilities of CA InterTest for CICS.

- From CICS, enter the transaction code **FILE**.
- From the Primary Option Menu, select Option 4 Auxiliary Storage.
- From any Source Listing display, breakpoint, or CORE display, enter **FILE** on the command line.

The following Auxiliary Storage Menu opens.

```
--------------CA InterTest r8.5  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):

    Protsym     p*______ ________ ________ ________ ________ ________ ________




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

To use the Auxiliary Storage menu, follow these steps:

1. Use the Option field to select the auxiliary storage type you want to view.
2. Optionally, enter one or more specific or generic names on the bottom fields of the menu. Generic names can include the masking characters * and +. Use ***** to indicate a string of any length, **+** to indicate a single character.
3. Press **Enter** to display a selection list of the files and queues that meet your selection criteria.

The File Selection List

Use the File Selection list, shown next, to select the specific file or queue you want to access. Type an **s** next to the File entry to select it for display.

Note: In a list spanning multiple pages, use the command LOCATE *filename* to position the display to the desired file.

```
-------------------- CA InterTest r8.5  FILE SELECTION----------------
 COMMAND ===>

   File     Data set name                                Attributes
 s PROTSYM  AD1DEV.INTERT60.PROTSYM                      Vsa, Ope, Ena, Rea
   *** End of data ***


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

For a list of the supported commands available from all CA InterTest for CICS Selection List displays and the standard PF Key definitions.

For more information:

Menus and Displays (see page 35)

Initial File Display

The Initial File Display screen is shown next.

```
DATATYPE= FC FILEID=          MODE=      LOG=OFF TODEST=         PASSWORD=

 FUNC=      SUBFUNC=      RETMETH=  ❶   ARGTYP=     SRCHTYP=

 MESSAGE=

  RETNRCID=                                                      CHGELEN=

     RCID=

     DATA=                                                       SIZE= 0000

 FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*

 LOC 0000  ........ ........ ........ ........  ................
                              ❷                                 ❸


                          CA InterTest for CICS r8.5

                 Copyright © 2014 CA.   All rights reserved



 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
```

This screen has three distinct areas:

❶

The header area contains fields for specifying:

- Type of data organization (DATATYPE)
- Record or segment to be retrieved
- Format in which the data is to be displayed (FORMAT)

The previous screen shows the default values for the DATATYPE field (FC - File Control) and for the FORMAT field (D - Dump).

❷

When FILE retrieves a record,it displays in this area.

❸

When FILE retrieves a record, it displays information about the file or PCB information for a DL/I database in this area. This area also displays the functions that can be performed, such as READ, BROWSE, and UPDATE etc against the FILE.

Note: FILE uses different screens for accessing and displaying DB2 and SQL/DS databases. For more information, see 2 later in this chapter.

Specify a File or Database

Follow this procedure to specify a file or a database:

1. Enter a file ID in the FILEID field, or the DL/I database identifiers in the PSB and DBD fields. No specification is needed for a DB2 or SQL/DS database.
2. If the default value in the DATATYPE field is not correct for the type of data organization you want to access, use PF6 to change it or replace the value with one of the following:

   FC for VSAM or BDAM files

   TS for Temporary Storage

   TD for Transient Data

   DL for DL/I databases

   DB for DB2 or SQL/DS databa

3. If the display format is not suitable, press PF2 until the format you want is displayed, or replace the value in the FORMAT field with one of the following:

   D for Dump format

   C for Character format

   V for Vertical format

   S for Structured format

   Each of these formats is described in the sections that follow.

4. If you do not want the display to start at the beginning of the record, change the value in the LOC field. Change this field at any time during the session.

Example

To view the data in the 150th byte of a record, specify 150 in the LOC field (95 for FORMAT= D) so byte 150 is the first byte displayed.

Dump Format

The following screen shows a record displayed in dump format.

On this screen:

- The left side of the record displays the hexadecimal representation of each byte
- The right side is the character representation of each displayable byte; non-displayable bytes are shown as periods

```
DATATYPE= FC FILEID= INVNTRY  MODE=BROWSE  LOG=OFF TODEST=   PASSWORD=
 FUNC= NEXT SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=
 MESSAGE= RECORD OBTAINED FOR VIEWING
  RETNRCID=F0F0F0F0F0F0F0F0F0F5                                  GELEN=
     RCID=
     DATA=                                                       SIZE= 0050
 FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*
 LOC 0000  ........ ........ ........ ........  ................
   0000    F0F0F0F0 F0F0F0F0 F0F54040 40404040   0000000005    DSORG=VSKS
   0010    F0F3F9F9 0407835C 40404040 40404040   0399....      RECFM=VB
   0020    40404040 40404040 C1C2C340 C3D6D9D7        BC CORP  LRECL=0050
   0030    F0F9F1F2 F8F74040 0000598C 40404040   091287  ....  BLKSIZE=0000
   0040    D1D6C8D5 404040E2 D4C9E3C8 40404040   JOHN   SMITH  KEYPOS=0000
   0050                                                       KEYLEN=0A
   0060                                                       STRNO=01
   0070
   0080                                                       READ
   0090                                                       BROWSE
   00A0
   00B0
   00C0

 1 Help       2 Format C   3 End        4 ENDB       5 NEXT    6 DataType DL
 7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom   12
```

Dump format has the following characteristics:

Record/segment display—16 bytes per line; hexadecimal and character.

Numeric information (such as, SIZE, LOC)—Hexadecimal.

Field entry—Hexadecimal or universal mode.

Position of first byte of data in record or segment—Location 0.

LOC Field

The LOC field determines where the display starts. For example, if the LOC field specifies 002F, the first byte of data displayed is the 48th byte of the record.

The line numbers under the LOC field and the scale to the right of the FORMAT field help you determine the location of any byte of data. Each line number indicates the first byte of data in that line.

To locate a particular byte, follow these steps:

1. Go to the correct line number.
2. Go across to the correct position. For example, to locate the 48th byte of data (X'2F') in the record in the previous figure, go to LOC line 0020.
3. Go across to position FF. This byte contains a 'P' (X'D7'ss).

Character Format

The following screen shows a record displayed in character format.

```
DATATYPE= FC FILEID= INVNTRY  MODE=BROWSELOG=OFF TODEST=      PASSWORD=
FUNC= NEXT SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=

MESSAGE= RECORD OBTAINED FOR VIEWING

 RETNRCID=0000000005                                          CHGELEN=
    RCID=

    DATA=                                                     SIZE= 0080
FORMAT= C 00000000011111111112222222222333333333344444444445
LOC 00001 12345678901234567890123456789012345678901234567890
          ..................................................  DSORG=VSKS
 00001    0000000005      0399....              ABC CORP09   RECFM=VB
 00051    1287  ....   JOHN   SMITH                          LRECL=0080
 00101                                                       BLKSIZE=00000
 00151                                                       KEYPOS=00001
 00201                                                       KEYLEN=010
 00251                                                       STRNO=001
 00301
 00351                                                       READ
 00401                                                       BROWSE
 00451
 00501
 00551

1 Help       2 Format V   3 End        4 ENDB    5 PREV   6 DataType DL
7 Page bwd   8 Page fwd   9 Caps Off  10 Top    11 Bottom 12
```

In this screen, the record displays in horizontal lines containing up to 50 characters per line and non-displayable bytes are represented by periods.

Character format has the following characteristics:

Record/segment display—50 characters per line

Numeric information (such as, SIZE, LOC)—decimal numbers

Field entry—character, decimal, or universal mode

Position of first byte of data in record or segment—LOCation 1

The LOC field determines where the display starts.

Example

If the LOC field specifies 0065, the first byte of data displayed is the 65th byte of the record.

The line numbers under the LOC field and the vertically arranged scale to the right of the FORMAT and LOC fields help you determine the position of any byte of data. Each line number indicates the first byte of data in that line.

To locate a particular byte, first go down to the correct line number, and then go across to the correct position.

Example

To locate the 65th byte of data in the record in the previous figure, go down to LOC line 00051, and then across to position 15. This byte contains a J.

Vertical Format

The following screen shows a record displayed in vertical format.

```
DATATYPE= FC FILEID= INVNTRY  MODE=BROWSELOG=OFF TODEST=         PASSWORD=
 FUNC= NEXT SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=

 MESSAGE=  RECORD OBTAINED FOR VIEWING
  RETNRCID=0000000005                                              CHGELEN=
     RCID=

     DATA=                                                         SIZE= 0080
 FORMAT= V 00000000011111111112222222222333333333344444444445
 LOC 00001 12345678901234567890123456789012345678901234567890
           ..................................................     DSORG=VSKS
  00001    0000000005      0399....               ABC CORP09     RECFM=VB
           FFFFFFFFFF444444FFFF00854444444444444444CCC4CDDDFF     LRECL=0080
           00000000050000000399473C000000000000000001230369709   BLKSIZE=00000
                                                                  KEYPOS=00001
  00051    1287  ....   JOHN   SMITH                             KEYLEN=010
           FFFF4400584444DDCD444EDCEC4444                         STRNO=001
           128700009C00001685000249380000
                                                                  READ
  00101                                                           BROWSE



 1 Help       2 Format D   3 End        4 ENDB       5 PREV       6 DataType DL
 7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

In vertical format, the data displays in groups of three lines similar to a DITTO output, with a blank line separating each group. Each line is represented as follows:

- Line 1 represents character format
- Line 2 represents hexadecimal values of the zone (high order) half-bytes
- Line 3 represents hexadecimal values of the digit (low order) half-bytes

Vertical format has the following characteristics:

Record/segment display—50 bytes per line; hexadecimal and character

Numeric data display (such as SIZE)—decimal numbers

Field entry— character, decimal, or universal mode

Position of first byte of data in record or segment—LOCation 1

The LOC field determines where the display begins.

Example

If the LOC field specifies 00020, the first byte of data displayed is the 20th byte of the record.

The line numbers under the LOC field and the vertically arranged scale to the right of the FORMAT and LOC fields help you determine the location of any byte of data. Each line number indicates the first byte of data in that line.

To locate a particular byte, first go down to the correct line number, and then go across to the correct position.

Example

To locate the 20th byte of data in the record in the previous figure, go down to LOC line 00001, and then across to position 20. This byte contains a '9' (X'F9').

Structured Format

FILE displays file, transient data, and temporary storage records or DL/I segments field-by-field in structured format. COBOL and PL/I structures use 01 level data names; Assembler structures use DSECT 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 for CICS 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 opens after the FORMAT field.

2. Identify the program and structure in the USE= field, as follows:

   USE=*symbolic-name*.*structure-name*

   symbolic-name

   Indicates the name of the program as defined in the Symbolic File.

   structure-name

   Indicates the COBOL or PL/I 01 level name or the Assembler DSECT.

3. Press Enter. CA InterTest for CICS displays the record in structured format.

Sample Records in Structured Format

The following screen shows a record in COBOL or PL/I structured format.

```
DATATYPE= FC FILEID= INVNTRY  MODE=      LOG=OFF TODEST=         PASSWORD=
 FUNC=      SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=
 MESSAGE= RECORD OBTAINED FOR VIEWING
  RETNRCID=F1D7D9D6E3E2E8D4400000000000000000000000000000000        CHGELEN=
     RCID= C'1'
     DATA=                                                          SIZE= 02D8
 FORMAT= S USE= PRODUCT.PRODREC
 LOC 0050           Name                      Hexadecimal            Character
 ------------------------------------------------------------------------------
 01 PRODREC                              |                          |
  02 PRODUCTTYPE                         | D3                       | L
  02 PRODUCTCODE                         | C2D4D5C8 40404040        | BMNH
  02 PRODUCTDESCRIPTION                  | E6D9C5D5 C3C8            | WRENCH  02
PRODUCTSITE                             | D1                       | J
                                        |                          |
                                        |                          |
                                        |                          |
                                        |                          |
                                        |                          |
                                        |                          |
 ------------------------------------------------------------------------------
-
 1 Help       2 Format D   3 End        4 BEGB       5            6 DataType DL
 7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

The following screen shows a record in Assembler structured format.

```
DATATYPE= FC FILEID= INVNTRY  MODE=      LOG=OFF TODEST=         PASSWORD=
FUNC=      SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=
MESSAGE=
 RETNRCID=D3C2D4D5C8404040404040404040404040404040404040404040        CHGELEN=
    RCID= C'1'
    DATA=                                                             SIZE= 0050
FORMAT= S USE= PRODUCT.PRODREC
LOC 0000  Displacement               Hexadecimal                     Character
-------------------------------------------------------------------------------
PRODTYPE             0 | D3                                        | L
PRODCODE             1 | C2D4D5C8 40404040                         | BMNH
PRODDESC             9 | E6D9C5D5 C3C8                             | WRENCH
PRODSITE            18 | D1                                        | J
                       |                                           |
                       |                                           |
                       |                                           |
                       |                                           |
                       |                                           |
                       |                                           |
                       |                                           |
 ------------------------------------------------------------------------------
1 Help       2 Format D   3 End        4 BEGB       5            6 DataType DL
7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

- Both screens list the contents of each data name in both hexadecimal and character format.
- For Assembler DSECTs, the display also includes the displacement.
- Non-displayable bytes are represented by periods.

Structured format is very similar to dump format:

Record/segment display—For Assembler, 16 bytes per line; for COBOL or PL/I, 12 bytes per line in hexadecimal and character

Numeric information (such as, SIZE, LOC)—Hexadecimal

Field entry—Hexadecimal or universal mode

Position of first byte of data in record or segment—LOCation 0

The structure-name remains valid throughout the FILE session unless it is changed. This lets you shift display formats or data types without re-entering the structure-name.

Special Features

The following bullets describe special features:

- To change the symbolic-name or structure-name, overtype the command in the USE= field.
- The FILE or PCB information is not displayed. To see this information, switch to another display format.
- The structure is positioned at the data name whose offset is less than or equal to the LOC= value, except during scrolling.

Use Structured Format When a Program Is at a Breakpoint

When a program is stopped at a breakpoint, substitute an asterisk (*****) for the symbolic-name.

Example

If program PAYROLL is stopped at a breakpoint and you enter:

```
USE=*.payrec
```

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

Use a Global Program Name

Your site compiles a default *dummy* COBOL or PL/I program containing *all* of the 01 level structures used at the site, or an Assembler program that contains all of the DSECTs. Symbolic information for this program must be stored in the CA InterTest for CICS Symbolic File.

The symbolic name for this program is referenced in the FSYMP installation name. The default name is PROTFILE.

Note: For details, see the *Installation Guide*.

If a default global program was compiled or assembled, obtain a structured record format by entering the following command:

```
USE=structure-name
```

The symbolic-name defaults to the default dummy program name.

Find a Data Name

To search the structure for a specific data name, enter the following command in the DATA= field:

```
FIND=data-name
```

Note: The FIND command is valid only when FORMAT=S.

data-name

Indicates the data name to be displayed.

The search is applied to the beginning of a data name.

Example

FIND=A9 finds the field named A9-AREA, but not the field named AREA-A9.

The search begins with the data name after the starting point and continues until either a match is found or the search has wrapped around to the original starting point. A message signals a possible wraparound.

Universal Mode of Data Entry in the RCID, DATA, and SSA Fields

Use the **universal mode** of entering data in the RCID, DATA, and SSA fields when you are entering information that contains both character and binary data (such as a VSAM key), or when it is inconvenient to change the display format.

Enter data as follows:

- **Character** Data. Specify C' (C and a single quote) before the character string and a ' (single quote) after it. Double any apostrophes that display within the string.

Example

C'O''ALLEY''

- **Hexadecimal** Data. Specify X' (X and a single quote) before the data, and a ' (single quote) after it. The data must be an even number of hexadecimal characters.

Example

X'001F'

Note: Specify an RCID up to 32-hexadecimal characters long.

You can also enter data that is both character and hexadecimal.

Example

C'1234'X'000F'

Access Files and Databases Protected by CA InterTest for CICS Passwords

Certain files or DL/I databases could have been assigned CA InterTest for CICS passwords when CA InterTest for CICS was installed. A password prevents unauthorized viewing and updating. FILE only lets you access such a file or database if you enter the correct password in the PASSWORD field. If you do not enter the correct password when one is required, FILE terminates.

Note: CA InterTest for CICS password protection decisions were made by the person who installed CA InterTest for CICS. For more information, consult that person or the *Installation Guide*.

Log FILE Session

FILE's logging facility lets you record the input commands and output screens generated during a FILE session in character format. The log is written to the transient data destination specified when CA InterTest for CICS was installed.

To see the name of the destination, follow these steps:

1. On a clear screen, enter **VRPT**.
2. Select option 02.
3. The GLOG option indicates the name of the destination.

To activate logging, enter:

```
FUNC= LOG
```

To terminate logging, enter:

```
FUNC= NLOG
```

When logging is in effect, the LOG field contains LOG= ON and the FUNC=COPY function is disabled. When it is not in effect, the LOG field contains LOG= OFF.

Note: If the AUDIT installation option was set at installation time, you cannot terminate logging.

End a FILE Session

Terminate FILE by pressing **PF3** or the Clear key.

PF Keys

When you are using FILE, the PF keys perform the functions listed in the following table.

PF Key	Function
PF1	Help initiates the CA InterTest for CICS online Help facility.
PF2	Format C V D S changes the format displayed to Character, Vertical, Dump, or Structured.
PF3	End or Clear terminates the FILE facility.
PF4	BEGB ENDB begins or ends a Browse for VSAM files specified for browse in the FILEID field.
PF5	PREV NEXT displays the previous or next record in browse mode.
PF6	DataType changes the DATATYPE to the code displayed next to the PF6 key: FC for File Control, DL for DL/I, TD for Transient Data, TS for Temporary Storage, and DB for both DB2 and SQL/DS.
PF7	Page Bwd scrolls the current display one page backward.
PF8	Page Fwd scrolls the current display one page forward.

PF Key	Function
PF9	Caps Off/Caps On sets the mode of translation when changing data in the character area of the display. When Caps On displays, all character data is accepted as typed. When Caps Off is displays, all character data on the line is translated to uppercase. Press PF9 to change from one mode to the other. **Note:** Translation to uppercase will not occur if an existing lowercase character would be affected; a message is issued instead.
PF10	Top scrolls the current display to the top of the work area.
PF11	Bottom scrolls the current display to the last page of the work area.

Scroll Through a Record

Use these PF keys to page forward and backward through a record, and to display the beginning or end of a record:

- PF8 pages forward
- PF7 pages backward
- PF10 displays beginning of record
- PF11 displays end of record

Use the LOC Field to Position the Display

The LOC field controls the position of the first displayed byte in the WORK AREA. The following table explains the values that you enter in the LOC field to control the position of the first displayed byte.

Important! To determine the location of any byte on the data display portion of the screen, add the number on the left end of the line (the number under the LOC field) to the number above the byte in the data display header (to the right of the FORMAT= field).

LOC	Position of Data
Hexadecimal value, relative to zero	For FORMAT=D or S, indicates the relative displacement of the first displayed byte.
Decimal value, relative to one	For FORMAT=C or FORMAT=V, indicates the relative displacement of the first displayed byte.

LOC	Position of Data
? and cursor is positioned in the record display area	The cursor position becomes the first displayed byte when you press Enter. The LOC field reflects the hexadecimal or decimal position, depending on the FORMAT= field.
FWD or PF8	Advances the data display *forward* one full page.
BWD or PF7	Moves the data display *backward* one full page.
TOP or PF10	Positions the display on the first byte.
END or PF11	Positions the display on the last page of data.

The Help Facility

The Help facility summarizes the steps required to perform FILE functions. Access the Help facility by pressing the PF1 key from any FILE screen. The following is the HELP screen that explains how to view a VSAM record.

```
              CA InterTest   INTERACTIVE HELP FACILITY
 TUTORIAL: TO VIEW A VSAM RECORD

 1. Start the FILE session, and specify:  DATATYPE= FC


 2. Specify the file name by entering:    FILEID= name of file


 3. Enter the following required fields:  FUNC= GET

                                          RCID= record identification


 4. Optionally enter one of the following values in the SRCHTYP or ARGTYP field:

             SRCHTYP= FKEQ  or  FKGE   (with RCID= a full key     KSDS only )

             SRCHTYP= GKEQ  or  GKGE   (with RCID= a generic key  KSDS only )

             ARGTYP= RBA               (with RCID= the 4 byte RBA of the record)

             ARGTYP= XRBA              (with RCID= the 8 byte RBA of the
record)

     NOTE : The following are the defaults:  ARGTYP=KEY and SRCHTYP=FKEQ


 5. Press ENTER to display the record data with the following fields filled in:


          SIZE= the number of bytes in the record (the size of the work area)

          RETNRCID= the ID of the returned record.                        (end)



 ENTER N FOR NEXT PAGE, P FOR PRECEDING PAGE, F FOR FIRST PAGE, OR

  M FOR RETURN TO PREVIOUS MENU. PRESS CLEAR TO EXIT.          SELECTION ==> M
```

Change Data in the Work Area

Change FILE work areas by overtyping the hexadecimal or character data displayed on the screen with new data. Any number of bytes can be overwritten with any uppercase or lowercase character.

Note: If you overtype the same data in hexadecimal and character display fields, the hexadecimal change takes precedence.

You can also replace record data with fill characters, data from CORE, or data from the saved WORK AREA.

The CHGE Function

Follow this procedure when using the CHGE function:

1. To change data in the WORK AREA, position the first byte (character) of the data that is to be changed so that it is the first byte of the FILE display. There are two ways to do this:
   - Type the desired location in the LOC field, and press Enter.
   - Type a question mark (?) in the LOC field, position the cursor at the appropriate location in the record, and press Enter.

   Delay pressing Enter until you complete Step 2.
2. Fill in the FUNC= and DATA= fields as shown next:
   ```
   FUNC=CHGE
   DATA= new data
   ```
3. Fill in the following *optional* field, if it applies to your situation:

   ```
   CHGELEN =  length of the data to be change
   ```

   Use a hexadecimal number for FORMAT=D and a decimal number for all other FORMATs.
4. Press Enter. The MESSAGE= field displays the following message and the displayed record shows the new data at the proper location.

   ```
   xxx NUMBER OF BYTES CHANGED
   ```

Example

To change four bytes at location 20 to the literal TEST, fill in the fields as shown next, and press Enter:

```
FUNC=CHGE
DATA=C'TEST'
LOC=00020                          (or 0013 for FORMAT=D)
```

Fill Part of the Work Area with a Character String

To fill part of the WORK AREA with a character string, specify the following fields, and press Enter:

```
FUNC=CHGE
SUBFUNC=FILL
CHGELEN= length of area to be change
DATA= character string
DOC= starting position
```

Example

To move 30 spaces into locations 15 to 44, specify the following fields and press Enter:

```
FUNC=CHGE
SUBFUNC=FILL
CHGELEN=0030            (or 001E for FORMAT=D)
DATA=C' '               (or X'40')
LOC=00015               (or 000E for FORMAT=D)
```

Copy Data from a Main Storage Location

To copy data from a main storage location into the WORK AREA, specify the following and press Enter:

```
FUNC=CHGE
SUBFUNC=CORE
DATA= CORE command that determines the source address
CHGELEN= length of area to be changed
LOC= starting position
```

Examples

The following examples demonstrate how to copy data from main storage.

1. To copy 16 bytes from TWA+8 (at a CA InterTest for CICS breakpoint) into locations 48to 63 of the record, specify the following, and press Enter:

```
FUNC=CHGE
SUBFUNC=CORE
CHGELEN=0016                (or 0010 for FORMAT=D)
DATA=CORE=TWA+8
LOC=00048           (or 002F for FORMAT=D)
```

2. Suppose you are stopped at a breakpoint in a COBOL program. You want to copy ten bytes of data from a field named SOC-SEC-NUMBER into locations 1 to 10 of the record. To do this, specify the following and press Enter:

```
FUNC=CHGE
SUBFUNC=CORE
CHGELEN=0010                          (or 000A for FORMAT=D)
DATA=CORE='SOC-SEC-NUMBER'
LOC=00001                   (or 0000 for FORMAT=D)
```

Copy Data from a Saved Work Area

To copy data from a saved work area into the current WORK AREA, specify the following, and press Enter:

```
FUNC=CHGE
SUBFUNC=FROM
CHGELEN= length of the area to be changed
DATA=FROMLOC= location of the data in the saved WORK AREA
LOC= starting position
```

Save Records and Display Work Areas

Save a copy of the displayed record in a WORK AREA. This is useful when you want to create new records using an existing record as a skeleton, or when you want to modify an existing record using data from a saved record.

To save the record currently displayed, enter:

```
FUNC= SAVE
```

A previously saved record area is destroyed.

To display the saved area, enter:

```
FUNC= DISS
```

To display the active WORK AREA, enter:

```
FUNC= DISW
```

Dump the Work Area

Write the work area to the CICS transaction dump file. To dump the WORK AREA, enter:

```
FUNC= DUMP
TODEST= a four character dump code
```

You can also enter an explanatory note in the MESSAGE field. The note displays in the dump.

Record a Copy of the Screen Display

Copy the WORK AREA to a transient data queue. To copy the work area to a TD queue, enter:

```
FUNC= COPY
TODEST= a transient data queue ID
```

You can also enter an explanatory note in the MESSAGE field. The note is written with the work area.

Note: This function is invalid when logging is in effect.

Common FILE Functions

Enter the following FILE commands in the FUNC= field for all data types except DB2 and SQL/DS databases.

Command	Function
ADDN	Acquires a new work area for record insertion.
ADDU	Uses an existing work area for record insertion.
CHGE	Changes data in the work area.
COPY	Copies a work area to the transient data destination specified in the TODEST field.
DISS	Displays the saved work area.
DISW	Displays the current work area.
DUMP	Dumps the work area using the dump code entered in the TODEST field.
END	Ends the FILE session.
HELP	Accesses Help for FILE.
LOG	Logs all FILE transaction requests and responses.
NLOG	Terminates logging.
SAVE	Creates a copy of the current work area.
SRCH	Searches for the data string specified in the DATA field (not valid for DL/I or transient data).

VSAM Files

The FILE transaction performs any VSAM function allowed in your CICS system.

Note: You might want to consult the *IBM CICS/VS Application Programmer's Reference Manual* for information on using VSAM files. This manual helps you formulate your FILE requests correctly, and explains the meaning of many responses you will receive from FILE.

To begin processing a VSAM file, first make sure you are viewing the correct screen. The DATATYPE= field should be set to FC. If not, press PF6 or enter:

```
DATATYPE= FC
```

Then enter:

```
FILEID= file name
```

The other information you enter depends on the function you want to perform.

Identify the Record

When you access a VSAM file for the first time, the record identification field is set as follows:

```
KSDS and ESDS    nulls    (X`00')
RRDSone (X`01')
```

Note: Data entered in the RCID field overrides these values.

Occasionally, the *last* record identification for a file is needed as the *first* record identification for another file. To prevent the RCID from being reset to nulls or to one, enter **LAST** in the SUBFUNC field when you specify the new filename in the FILEID field. The LAST subfunction works only when a file is requested for the first time.

FILE Screen Layout for VSAM Files

The following screen shows a blank header area on a FILE screen for VSAM files.

Examples

Examples of the complete screen, see Dump Format (see page 447), Character Format (see page 449), Vertical Format (see page 450) and Structured Format (see page 451) earlier in this chapter.

```
DATATYPE= FC FILEID=          MODE=     LOG=     TODEST=          PASSWORD=
FUNC=      SUBFUNC=      RETMETH=        ARGTYP=     SRCHTYP=
MESSAGE=
 RETNRCID=                                                        CHGELEN=
    RCID=

    DATA=                                                         SIZE=
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*
LOC 0000  ........ ........ ........ ........  ................




1 Help       2 Format D   3 End        4 BEGB       5            6 DataType DL
7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

The fields provide the following information:

Fields	Function
DATATYPE	Identifies the type of data organization. This must be FC.
FILEID	Specifies the name of the VSAM file.
MODE	Identifies the FILE operation such as browse, update, add, add in mass insertion. This field is blank if you are just viewing a record.
LOG	Specifies whether the FILE logging facility is on or off.
TODEST	Identifies a transient data destination (when a work area is copied) or a dump identification (when a work area is dumped).
PASSWORD	Must contain a four-character password if the file is protected.
FUNC	Specifies the FILE function. When this field contains a function code, it is executed if you press Enter. In addition to common FILE functions, enter the following commands: BEGB, DEL, ENDB, GET, GETU, NEXT, PREV, PUT, REL.
SUBFUNC	Modifies a FUNC specification.
RETMETH	Identifies the RCID field (BDAM files only).

Fields	Function
ARGTYP	No longer required.
SRCHTYP	Specifies how RCID information is processed (KSDS files only).
MESSAGE	Contains FILE messages. Also specify messages here when FUNC=COPY or FUNC=DUMP.
RETNRCID	Contains the record key of the retrieved record.
CHGELEN	Specifies the number of bytes changed in the work area when FUNC=CHGE.
RCID	Identifies the record to be processed.
DATA	Contains data needed to perform certain FILE functions.
SIZE	Specifies the record size or the number of records to be searched.
FORMAT	Specifies the format in which the record is displayed.
USE	Specifies the program and structure for structured format.
LOC	Specifies a location within a record.

Some fields are used only for certain tasks. The discussion of each FILE function indicates which fields must be specified.

When a record displays, its FILE information is in the lower-right portion of the screen. This information includes the following:

- DSORG(data organization)
- RECFM(record format)
- LRECL (record length)
- BLKSIZE (blocksize)
- KEYPOS (key position)
- KEYLEN (key length)
- STRNO (string number)

Beneath the FILE information is a list of the functions you can perform, such as READ, ADD, UPDATE, BROWSE, DELETE.

FILE Functions for VSAM Records

The following FILE functions are discussed in this section:

- Viewing a VSAM record
- Browsing a VSAM file
- Searching for a data string
- Updating a VSAM record
- Adding a VSAM record
- Mass insertion into a VSAM file
- Copying a VSAM record from one file to another
- Deleting VSAM records

View a VSAM Record

To view a VSAM record, specify the READ operation on the FILE definition.

1. Enter the following required information for all VSAM files:

   ```
   FUNC= GET
   RCID= key
   ```

   For a KSDS file, the key is a full or generic key, depending on the optional SRCHTYP field. Specify the key in hexadecimal or character format or a combination of both.

   For an ESDS file, the key is a four-byte hexadecimal number representing the relative byte address (RBA).

   For an RRDS file, the key is a four-byte hexadecimal number representing the relative record number.

2. Enter the following optional information for a KSDS file:

   ```
   SRCHTYP= FKEQ/FKGE/GKEQ/GKGE
   ```

- If you specify SRCHTYP=FKEQ, enter the full key in the RCID field. FILE retrieves the specified record. This is the default.
- If you specify SRCHTYP=FKGE, enter the full key in the RCID field. FILE retrieves the specified record or the record with the next higher key.
- If you specify SRCHTYP=GKEQ, enter a generic (partial) key in the RCID field. FILE retrieves the first record whose key matches the generic key.
- If you specify SRCHTYP=GKGE, enter a generic (partial) key in the RCID field. FILE retrieves the first record whose key matches the generic key or the next higher key.
- Press Enter. FILE displays the requested record.

Browse a VSAM File

To browse a VSAM file, specify the BROWSE operation on the FILE definition.

1. Enter the following required information for all VSAM files (or press PF4):

   FUNC= BEGB

   Also specify a key in the RCID field to begin to browse from a specified record.

   - For a KSDS file, this key can be a full or generic key, depending on the SRCHTYP field.
   - For an ESDS file, you must also specify ARGTYP= RBA or ARGTYP= XRBA.

2. Enter the following optional information for a KSDS file:

   SRCHTYP= FKGE/FKEQ/GKGE/GKEQ

   - If you specify SRCHTYP=FKGE, enter the full key in the RCID field. The browse starts at the record with that key or the next higher key. This is the default.
   - If you specify SRCHTYP=FKEQ, enter the full key in the RCID field. The browse starts at this record.
   - If you specify SRCHTYP=GKGE, enter a generic (partial) key in the RCID field. The browse starts at the first record whose key matches the generic key or the next higher key.
   - If you specify SRCHTYP=GKEQ, enter a generic (partial) key in the RCID field. The browse starts at the first record whose key matches the generic key.

3. Press Enter. FILE displays the first retrieved record. The FUNC field is preset to NEXT.

   - To view the records in ascending sequence, press Enter.
   - To view the records in descending sequence, enter FUNC=PREV, and then repeatedly press Enter to display the records in descending sequence. Use the PREV function only if the browse is initiated with SRCHTYP=FKEQ.
   - Use both NEXT and PREV during the same browse operation. Use PF5 to change descending and ascending sequence.

4. Terminate the browse by pressing PF4, specifying FUNC=ENDB, or specifying one of these functions: GET, GETU, ADD, or ADDU.

Example

To view the record with the key 'REC001' and the next record, and then to terminate the browse, follow these steps:

1. Specify the following (or press PF4):

   ```
   FUNC= BEGB
   RCID= C`REC001'
   ```

   Press Enter to display the record with a key of REC001.

2. Press Enter to display the next record.

3. Press PF4 to end the browse.

Search for a Data String

To search for a data string, specify the BROWSE operation on the FILE definition.

1. Specify the following required information:

   ```
   FUNC= SRCH
   RCID= id of record where search should begin
   DATA= data string to be located
   LOC a number or the keyword ANY
   ```

   If a number is entered in the LOC field, FILE looks for the data string only at the specified location. If ANY is entered, FILE searches the entire record for the specified data string.

2. To search more than one record, enter the number of records that you want to search in the SIZE field.

3. Specify the following optional information for a KSDS file:

   SRCHTYP= FKGE/FKEQ/GKEQ/GKGE

   - If you specify SRCHTYP=FKGE, enter the full key in the RCID field. The search starts with the specified record or the record with the next higher key. This is the default.
   - If you specify SRCHTYP=FKEQ, enter the full key in the RCID field. The search starts with the specified record.
   - If you specify SRCHTYP=GKEQ, enter a generic (partial) key in the RCID field. The search starts with the first record whose key matches the generic key.
   - If you specify SRCHTYP=GKGE, enter a generic (partial) key in the RCID field. The search starts with the first record whose key matches the generic key or the next higher key.

4. Press Enter. If FILE finds the requested data string, it displays the record in which it is found, starting with the requested string. The RETNRCID field displays the ID of the record and the LOC field displays the record position at which the data was found.

   If the data is not found, the message DATA NOT FOUND opens.

Example

To scan up to ten records in the file ACCTS looking for the data string TEST in record positions 20-23, and to begin the search with the record whose ID is NEWREC001, specify the following:

```
FILEID= ACCTS
FUNC= SRCH
RCID= C`NEWREC001'
DATA= C`TEST'
LOC 00020               (or 0013 for FORMAT=D)
SIZE= 0010              (or 000A for FORMAT=D)
```

Press Enter to display the data.

Update a VSAM Record

To update a record, specify the UPDATE operation on the FILE definition.

1. Retrieve the record you want to update, and then enter the appropriate command:

   ```
   FUNC= GETU
   RCID= key            (for all VSAM files)
   FUNC= GET
   RCID= key            (for KSDS files only)
   ```

   Press Enter, or browse the file.

2. Change any portion of the record, except a KSDS key. For details, see Change Data in the Work Area.
3. After you have made all the changes, specify the following, and then press Enter:

   ```
   FUNC= PUT
   SIZE= new record size
   ```

   Specify the SIZE field only when changing the size of a variable length record.
4. FILE rewrites the updated record.

Add a VSAM Record

To add a record, specify the ADD operation on the FILE definition.

1. Specify the following required information:

   ```
   FUNC= ADDN  (to obtain a new work area)
   ```

   or

   ```
   FUNC=ADDU   (to use the data in the current work area)
   ```

   Press Enter to get the work area to be used for the new record. FILE displays the maximum record length in the SIZE field and ADD in the MODE field.
2. Specify the following optional information:

   ```
   FUNC=CHGE
   SUBFUNC=FILL
   DATA=data string
   ```

   Press Enter.

   - For FUNC=ADDN, the data string fills the new work area.
   - For FUNC=ADDU, the data string fills the extended portion of the work area.
3. Change the data in the work area to meet your requirements. See Change Data in the Work Area for details.
4. To write the updated record, enter the following required information:

   ```
   FUNC= PUT
   RCID= record id
   SIZE= record length
   ```

   For a KSDS file, the record ID overlays the key portion of the record.

   Note: Specify the SIZE field only when adding a variable length record. Press Enter. FILE redisplays the new record and displays the message RECORD ADDED.
5. If you decide not to add this record, specify FUNC= REL. FILE continues to display the record and displays the message FWA HAS BEEN RELEASED.

Mass Insertion into a VSAM File

Follow the same procedure for adding a single record.

1. Specify FUNC=ADDN, or FUNC=ADDU and SUBFUNC=MASS. Optionally specify the DATA= data string field.
2. Make your changes and write the new record. Repeat the procedure of making changes and writing the new record until you have added all the records.
3. Enter FUNC= REL to terminate the mass insertion operation.

Example

1. Specify the following, and then press Enter:

```
FILEID= ACCTS
FUNC= ADDN
SUBFUNC= MASS
```

2. Specify the following, and then press Enter:

```
FUNC= CHGE SUB
FUNC= FILL DATA= C' '
```

3. Specify the following, and then press Enter:

```
FUNC= PUT
RCID= C'NEWREC004'
```

4. Specify the following, and then press Enter:

```
FUNC= PUT
RCID= C'NEWREC005'
```

5. Specify the following, and then press Enter:

```
FUNC= PUT
RCID= C`NEWREC008'
```

6. Specify the following, and then press Enter:

```
FUNC= REL
```

Copy a VSAM Record from One File to Another

To copy a record from one file to a second file, specify the READ function for the first file and the ADD function for the second file.

1. To retrieve the record you want to copy, specify the following, and then press Enter or browse the file:

   ```
   FILEID= file ID of source file
   FUNC= GET
   RCID= key
   ```

2. After FILE displays the record, specify the following, and then press Enter:

   ```
   FILEID= file ID of destination file
   FUNC= ADDU
   ```

3. Make any necessary changes, then specify the following, and press Enter:

   ```
   FUNC= PUT
   RCID= key
   SIZE= record length
   ```

   Note: Specify the SIZE field only when adding a variable length record.

   FILE redisplays the record and displays the message RECORD ADDED.

   Repeat Steps 1 through 3 to copy additional records.

   Note: If the LRECLs of the two files differ, the LRECL of the destination file determines the size of the record. The record will either be truncated or padded with binary zeros to meet that length.

Example

To copy a record with the key 'REC050' from the ACCTS file to the TEST file, follow these steps:

1. Specify the following, and then press Enter to display the record:

   ```
   FILEID= ACCTS
   FUNC= GET RCID= C'REC050
   ```

2. Specify the following, and then press Enter:

   ```
   FILEID= TEST
   FUNC= ADDU
   ```

3. Specify the following, and then press Enter:

   ```
   FUNC= PUT
   RCID= C'REC001'
   ```

Delete VSAM Records

To delete a record, follow these steps:

1. Specify the DELETE operation on the FILE definition.
2. Specify the following required information:

```
FUNC= DEL
```

To delete a *single record*, follow these steps:

1. Specify the following:

```
RCID= full record key
```

2. Press **Enter.**

To delete *all records* having a generic (partial) key (KSDS files only), follow these steps:

1. Specify the following:

```
RCID= generic key
SRCHTYP= GKEQ
ARGTYP= KEY
```

2. Press **Enter**.

Notes:

- FILE deletes all the records whose keys match the generic key *unless* BACKUPTYPE DYNAMIC is specified for the FILE definition because CICS does not permit generic key deletion for FILE defined with this option
- If a single record is deleted, FILE displays it with the message RECORD DELETED.
- If you specify SRCHTYP=GKEQ, FILE displays the first record deleted. The message field specifies the number of deleted records xxx RECORDS DELETED.

Examples

1. To delete the record NEWREC001 from the ACCTS file, specify the following, and then press Enter:

```
FILEID= ACCTS
FUNC= DEL
RCID= C`NEWREC001'
```

2. To delete all records whose keys start with the character string NEWREC from the ACCTS file, specify the following, and then press Enter:

```
FILEID= ACCTS
FUNC= DEL
SRCHTYP= GKEQ
RCID= C`NEWREC'
ARGTYP= KEY
```

Deleting a Record with a Large Key

To delete a record whose key is larger than the RCID field, perform these steps:

To retrieve the record or browse the file, follow these steps.

1. Specify the following:

   ```
   FUNC= GET
   SRCHTYP= GKGE
   RCID= generic key
   ARGTYP= KEY
   ```

2. Press **Enter**.

To delete the record, follow these steps.

1. Specify the following:

   ```
   FUNC= PUT
   SUBFUNC= DEL
   ```

2. Press **Enter**.

   FILE deletes the record and displays the message RECORD DELETED.

Work with DL/I Databases

FILE lets you access any DL/I database to which it has been given access.

To start working with a DL/I database follow these steps.

1. Initiate the FILE transaction and get the DL/I entry screen.
2. Press **PF6** or specify the following:

   ```
   DATATYPE= DL
   ```

3. Then enter:

   ```
   PSB= PSB name
   DBD= DBD name
   NO= DBD entry number
   ```

Other information you enter depends on the function you want to perform.

Note: For more information about using DL/I databases, see *IBM's DL/I DOS/VS Application Programming Reference Manual* or the *IMS/VS Application Programming for CICS/VS Users Manual.*

FILE Screen Layout for DL/I Databases

The following screen shows the FILE screen layout for DL/I databases.

```
DATATYPE= DL FUNC=      PSB=          DBD=          NO 01 SUBFUNC=      PS=
 SSA=

                                                                    TODEST=
MESSAGE=                                                            CHGLEN=
    DATA=                                                           SIZE= 0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*  LOG=ON
LOC 0000  ........ ........ ........ ........   ................
                                                                    STAT=
                                                                    NAME=
                                                                    LEVL=
                                                                    POPT=
                          CA InterTest for CICS r8.5                  KFBL=

                                                                    KEY FEEDBACK
                Copyright (c) 2014 CA.  All rights reserved.
                                                                    *          *
                                                                    *          *
                                                                    *          *
                                                                    *          *
                                                                    *          *
                                                                    *          *
--------------------------------------------------------------------------------
1 Help       2 Format C   3 End        4 BEGB       5            6 DataType FC
7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

The following table describes each field's function.

Field	Function
DATATYPE	Specifies the type of data organization. For DL/I databases, it is DL.
FUNC	Specifies the FILE function. When this field contains a function code, it is executed if you press Enter. In addition to common FILE functions, enter the following commands: DLET, GHN, GHU, GN, GNHP, GNP, GU, ISRT, REPL.
PSB	Specifies the PSB name, which you can change any time using FILE.
DBD	Specifies the DBD name, which you can change any time using FILE.
NO	Specifies which DBD entry to use when there is more than one DBD with the same name in the PSB. The default is 01.
SUBFUNC	Modifies a FUNC specification.
PS	If the DBD is password protected, enter a four-character password.
SSA	Specifies required segment search arguments. See the section SSA Formats.

Field	Function
TODEST	Identifies a transient data destination (when a work area is copied) or a dump identification (when a work area is dumped).
MESSAGE	Contains FILE messages. Also specify messages here when FUNC=COPY or FUNC=DUMP.
CHGELEN	Specifies the number of bytes changed in the work area when FUNC=CHGE.
DATA	Contains data needed to perform certain FILE functions.
SIZE	Specifies the size of the segment.
FORMAT	Specifies the format in which the segment is displayed.
USE	Specifies the program and structure for structured format.
LOG	Specifies whether FILE's logging facility is on or off.
LOC	Specifies a location within a segment.

Returned PCB information appears in the lower-right screen area:

STAT—DL/I status code returned by PSB

NAME—Name of retrieved segment

LEVL— Level number of segment

POPT—DL/I processing options

KFBL—Length of key feedback area

KEY FEEDBACK—Displays key feedback area

Some fields are applicable only to certain functions. The discussion of each FILE function indicates which fields must be specified.

Specifying the SIZE Field

For z/OS CICS 4.1 users:

Use the SIZE field to specify a storage area large enough to contain the segment to be retrieved or added. If this field is blank or contains zeros, FILE uses the default value specified in the DL1SIZE option when CA InterTest for CICS was installed. To check the default value:

1. On a clear screen, enter **VRPT**.
2. Select option 02.
3. The DL1SIZE option contains the default SIZE value.

If you need to change the default value, contact the systems programmer who maintains CA InterTest for CICS.

FILE Functions for DL/I Databases

The following topics are discussed in this section:

- SSA formats
- Viewing a DL/I segment
- Updating a DL/I segment
- Adding a DL/I segment
- Copying a DL/I segment
- Deleting a DL/I segment

SSA Formats

SSA specifications follow standard DL/I conventions. Information entered in the SSA field must conform to the following format:

```
segname qual keyname ro keyarg
```

segname

Indicates the segment name, from one to eight characters long. If the segment name is less than eight characters, it must be followed by a blank.

qual

Indicates a blank, left parenthesis, (, or asterisk, *.

- Blanks specify an unqualified SSA.
- A left parenthesis, (, specifies a qualified SSA. If used, right parenthesis,), must follow the *keyarg* data.
- An asterisk, *, specifies that command codes are present. In this case, either a blank (for an unqualified SSA) or a left parenthesis (for a qualified SSA) must follow the last command code. FILE does *not* validate command codes. This is handled by DL/I.

keyname

Indicates the key name, from one to eight characters long. If it is less than eight characters long, it must be followed by one blank.

ro

Indicates a relational operator. It can be any of the standard relational operators used in DL/I. It must be two bytes long and must immediately follow either the last character in the keyname (if the keyname is eight characters long), or the blank terminating the keyname (if the keyname is less than eight characters long).

If you use a one-character sign rather than two letters to specify the relational operator (such as = instead of EQ), you must specify a blank before or after the sign.

keyarg

Indicates the argument for the key name. Enter the keyarg as either character or hexadecimal data, or as a combination of both, in one of the following ways:

- C'data' for character data
- X'data' for hexadecimal data
- data for treated as character data

  Note: If your *keyarg* data contains any of these special characters,), &, |, *, or + you must use the X'data' format to enter it.
- Create compound conditions by using & or | signs to join multiple *keyname ro keyarg* specifications.

  Note: If you specify more than one SSA, you must insert a blank and SSA= before each subsequent SSA.

Examples

In the following examples of SSA specifications, b indicates a blank.

1. The following specification indicates an unqualified SSA:

   ```
   SSA= SEGTWObb
   ```

   The two blanks are necessary, because the segment name is less than eight characters, and an unqualified SSA is indicated.

2. The following specifications indicate a qualified SSA:

   ```
   SSA= EMPLOYEE(NAMEb=bSMITH)
   SSA= EMPLOYEE(NAMEbEQSMITH)
   ```

3. The following specification indicates an unqualified SSA with the command code L:

   ```
   SSA= CARSb*Lb
   ```

4. The following specification indicates a qualified SSA with a compound condition:

   ```
   SSA= EMPLOYEE(NAMEb=bSMITH&CODEbLTX'FFF0')
   ```

View a DL/I Segment

Follow this procedure to view a DL/I segment:

1. Specify the following required information:

   ```
   FUNC= a DL/I get function
   SIZE= size of DL/I segment
   ```

2. Enter the following optional information:

   ```
   SSA= the SSAs needed to retrieve the requested segment
   ```

3. Press **Enter**. The returned PCB information is displayed in the lower-right portion of the FILE screen. If the call is successful, the segment also displays.

Examples

1. To retrieve the first root segment, which is 975 bytes long, specify the following, and then press Enter:

   ```
   FUNC= GU
   SIZE= 975          (or 3CF for FORMAT= D)
   ```

2. To sequentially process a database whose longest segment is 2000 bytes, specify the following, and then press Enter for each segment:

   ```
   FUNC= GN
   SIZE= 2000         (or 7D0 for FORMAT= D)
   ```

3. To retrieve an EMPLOYEE segment, which is 100 bytes long, specify the following, and then press Enter. (Its key is ADAMS, and its parent is a SKILL segment with a key of ARTIST.)

   ```
   FUNC= GU
   SIZE= 100          (or 64 for FORMAT= D)
   SSA= SKILL (SKILCODE= ARTIST) SSA=EMPLOYEE(NAME = ADAMS)
   ```

Update a DL/I Segment

Follow this procedure to update a DL/I segment:

1. Specify the following required information:

   ```
   FUNC= GHU, GHN, or GHNP
   ```

2. Enter the following optional information:

   ```
   SIZE= size of DL/I segment
   SSA= the SSAs needed to retrieve the requested segment
   ```

3. Press **Enter**. The returned PCB information is displayed in the lower-right portion of the FILE screen. If the call is successful, the segment is also displayed. The STAT field indicates if the call is successful. Make your changes to the displayed segment. See the section Change Data in the Work Area for details.
4. To rewrite the updated segment, specify the following**:**

   ```
   FUNC= REPL
   ```

5. Press **Enter**.

   The returned PCB information is displayed.

Example

To retrieve, change, and replace a SKILL segment whose key is ARTIST and whose length is 100 bytes, follow these steps:

1. Specify the following, and then press **Enter**:

   ```
   FUNC= GHU
   SIZE= 100           (or 64 for FORMAT= D)
   SSA= SKILL   (SKILCODE= ARTIST)
   ```

2. Change the segment.
3. Specify the following**:**

   ```
   FUNC= REPL
   ```

4. Press **Enter**.

Adding a DL/I Segment

Follow this procedure to add a DL/I segment:

1. Specify the following required information:

   ```
   FUNC= ADDN
   SIZE= size of the DL/I segment
   ```

2. Enter the following optional information:

   ```
   DATA= data string to initialize the new segment
   ```

3. Press **Enter** to display the work area for the new segment. FILE displays the message WORK AREA OBTAINED.
4. Change the displayed segment. For details, see Change Data in the Work Area earlier in this chapter.
5. To write the new segment, enter the following required information, and then press **Enter**:

```
FUNC= ISRT
SSA= SSAs necessary to insert segment into proper
     position in database
```

The returned PCB information is displayed in the lower-right portion of the screen. The segment is added if the call is successful.

Example

To add a new 100 byte SKILL segment whose key is ENGINEER, follow these steps:

1. Specify the following, and then press **Enter:**

```
FUNC= ADDN
SIZE= 100
```

2. Change the segment.
3. Specify the following**:**

```
FUNC= ISRT
SSA= SKILL (SKILCODE= ENGINEER)
```

4. Press **Enter**.

Copy a DL/I Segment

Copy a DL/I segment from one database to another.

1. To retrieve the segment you want to copy, specify the following, and then press **Enter:**

```
FUNC= a DL/I get function
PSB and DBD of source database
SIZE= size of the DL/I segment
```

2. Enter the following optional information, and then press **Enter**:

```
SSA= SSAs needed to retrieve the requested segmentFILE displays the segment.
```

3. Specify the following, and then press **Enter**:

```
FUNC= ADDU
PSB and/or DBD of destination database
SIZE= size of the DL/I segment
```

4. Make any necessary changes.
5. Specify the following, and then press **Enter**:

```
FUNC= ISRT
SSA= SSAs necessary to insert segment into proper
     position in database
```

The segment is copied if the call is successful.

Note: To copy additional segments, repeat Steps 1 through 5.

Example

To copy a SKILL segment whose key is ARTIST and whose length is 100 bytes from one database to another, follow these steps:

1. Specify the following, and then press **Enter**:

```
FUNC= GU
PSB= OLDBASE
DBD= DBD0025
SIZE= 100              (or 64 for FORMAT= D)
SSA= SKILL (SKILCODE= ARTIST)
```

2. Specify the following, and then press **Enter**:

```
FUNC= ADDU
PSB= NEWBASE
DBD= DBD0010
SIZE= 100              (or 64 for FORMAT= D)
```

3. Specify the following, and then press **Enter**:

```
FUNC= ISRT
SSA= SKILL (SKILCODE= ARTIST)
```

Delete DL/I Segments

Follow this procedure to delete a DL/I segment:

Note: To avoid damage to the integrity of the database, take the same precautions that you take when you program in DL/I.

1. Enter the following required information:

   ```
   FUNC= GHU, GHN, or GHNP
   SIZE= size of the DL/I segment
   ```

2. Enter the following optional information:

   ```
   SSA= the SSAs needed to retrieve the requested segment
   ```

3. Press **Enter**. The returned PCB information is displayed in the lower-right area of the FILE screen. If the call is successful, the segment is also displayed. The STAT field indicates if the call is successful.
4. To delete the retrieved segment, enter:
5. FUNC= DLET
6. Press **Enter**. The returned PCB information is displayed.

Example

To delete the SKILL segment whose key is ARTIST, follow these steps:

1. Specify the following:

   ```
   FUNC= GHU SSA= SKILL (SKILCODE= ARTIST)
   ```

2. Press **Enter**.
3. Specify the following:

   ```
   FUNC= DLET
   ```

4. Press **Enter**.

DB2 and SQL/DS Databases

FILE lets you access DB2 or SQL/DS databases to which it has access.

Note: For more information, see *IBM's DATABASE2 Application Programming Guide for CICS/OS/VS Users* or *IBM's SQL/Data System Application Programming Guide.*

Access the DB2 Facility

To start working with a DB2 database, select one of the following options:

- From a clear CICS screen, enter **FILE**.
- On the file display, change the data type to **DB**.
- From a clear CICS screen, enter **ITST**.
- On the Auxiliary Storage Menu, select Option 4.
- On the resulting menu, select Option 2, DB Database.

The following screen opens.

```
                        CA InterTest    File DB2 Facility

EXEC SQL

         ------------------------------------------------------

         ------------------------------------------------------


                     Enter the SQL command to be executed.

_____________________________________________________________________________
1 Help       2 CMND list 3 END       4 Large CMND  5 Log On     6 DataType FC
7            8           9          10            11           12
```

SQL Commands

Enter Structured Query Language (SQL) commands to perform DB2 or SQL/DS functions. FILE accepts the following SQL commands, subject to user authorization: SELECT, UPDATE, DELETE, INSERT, CREATE, ALTER, DROP, LABEL ON, GRANT, REVOKE, and COMMENT ON.

Note: For more information, see *the IBM DATABASE2 Application Programming Guide for CICS/OS/VS Users* or *SQL/Data System Application Programming Guide.*

Command Rules

Observe the following rules in entering SQL commands:

- Begin the command in column one.
- Enter only one command at a time.
- A command can exceed several lines. If the command does not fit in the allowed space, press **PF4** to obtain a larger input area. The command remains in the large input area for redisplay (using PF4), modification, and execution.

Scroll Through the DB2 and SQL/DS Table Display

FILE displays DB2 and SQL/DS table data horizontally in columns and vertically in rows.

- The columns represent the fields in the database
- The rows represent the records for each member in the table

The following screen shows an SQL table data display.

```
CA InterTest   File DB2 Facility
EXEC SQL                       # Cols 012   Max Loc 00141   Loc 00001 Page 001

SELECT * FROM DSN8210.EMP


1-------2------------3---------4---------------5----------6----------7------
 EMPNO  | FIRSTNME    | MIDINIT | LASTNAME       | WORKDEPT | *PHONENO | *HIRE

--------+-------------+---------+----------------+----------+----------+-----+

 000210 | WILLIAM     | T       | JONES          | D11      | 0942     | 1979
 000220 | JENNIFER    | K       | LUTZ           | D11      | 0672     | 1968
 000070 | EVA         | D       | PULASKI        | D21      | 7831     | 1980
 000230 | JAMES       | J       | JEFFERSON      | D21      | 4265     | 1966
 000240 | SALVATORE   | M       | MARINO         | D21      | 3780     | 1979
 000250 | DANIEL      | S       | SMITH          | D21      | 0961     | 1969
 000260 | SYBIL       | V       | JOHNSON        | D21      | 8953     | 1975
 000270 | MARIA       | L       | PEREZ          | D21      | 9001     | 1980
 000050 | JOHN        | B       | GEYER          | E01      | 6789     | 1949
 000090 | EILEEN      | W       | HENDERSON      | E11      | 5498     | 1970
 000280 | ETHEL       | R       | SCHNEIDER      | E11      | 8997     | 1967
 000290 | JOHN        | R       | PARKER         | E11      | 4502     | 1980
 000300 | PHILIP      | X       | SMITH          | E11      | 2095     | 1972

1 Help       2 CMND list 3 END       4 Large CMND  5 Log On     6 DataType FC

7 Page bwd   8 Page fwd  9 Page 1   10 Scrl. -->  11 Scrl. <-- 12 Structure
```

In the previous screen, the column headings (EMPNO, FIRSTNME, MIDINIT, and so on) represent all fields in the table. A heading preceded by an asterisk indicates that the field contains null characters (for example, *PHONENO).

The vertical rows represent the members in the table. In this example, each member contains information for a different employee. If all the rows cannot display on one screen, press **PF8** to page down through the table. Press **PF7** to page up again.

FILE provides the following information above the screen display:

 # Rows—specifies the total number of rows in the display. FILE displays this field only if you page down to the end of the table.

Cols—specifies the number of columns defined in the table.

Max Loc—specifies the horizontal width of the screen display.

Loc—specifies the position of the leftmost character on display.

Page—specifies the page you are currently viewing.

Display Additional Columns

If all the columns cannot display at one time, shift the display in one of the following ways:

- Tab to the line that separates the displayed columns and press **Enter**. The position indicated by the cursor becomes the first position on your screen.
- Place the cursor under any character in the column heading and press **Enter**. The position indicated by the cursor becomes the first position on your screen.
- Change the value in the Loc field by entering *#nn* (clear any remaining numbers), and press **Enter**.

Example

If you enter #10 in the Loc field, the display starts with the first character in column 10.

- Change the value in the Loc field and press Enter.E

Example

If you enter 50 in the Loc field, the display starts with the 50th character.

- Press **PF11** to shift the display 80 characters to the left. Pressing **PF10** shifts the display back 80 characters to the right.

Example

In the previous display, the Loc field indicates that the display begins with position 1; the Max Loc field indicates that the entire table display is 141 characters wide. To see the remaining columns, you must scroll the display. If you tab the cursor to the line preceding HIREDATE and press **Enter**, FILE displays the following screen:

```
                        CA InterTest   File DB2 Facility        ➘
EXEC SQL  # Rows 00032          # Cols 012   Max Loc 00141   Loc 00075 Page 002

SELECT * FROM DSN8210.EMP


7------------8---------9---------10---11-----------12------
|*HIREDATE |*JOBCODE |*EDUCLVL |*SEX |*BRTHDATE   |*SALARY |
+----------+---------+---------+-----+----------+----------+
| 19790411 | 52      | 17      | M   | 19530223 | 18270.00 |
| 19680829 | 55      | 18      | F   | 19480319 | 29840.00 |
| 19800930 | 56      | 16      | F   | 19530526 | 36170.00 |
| 19661121 | 53      | 14      | M   | 19350530 | 22180.00 |
| 19791205 | 55      | 17      | M   | 19540331 | 28760.00 |
| 19691030 | 52      | 15      | M   | 19391112 | 19180.00 |
| 19750911 | 52      | 16      | F   | 19361005 | 17250.00 |
| 19800930 | 55      | 15      | F   | 19530526 | 27380.00 |
| 19490817 | 58      | 16      | M   | 19250915 | 40175.00 |
| 19700815 | 55      | 16      | F   | 19410515 | 29750.00 |
| 19670324 | 54      | 17      | F   | 19360328 | 26250.00 |
| 19800530 | 42      | 12      | M   | 19460709 | 15340.00 |
| 19720619 | 48      | 14      | M   | 19361027 | 17750.00 |
| 19640912 | 43      | 12      | F   | 19310421 | 15900.00 |
_____________________________________________________________________________

1 Help       2 CMND list 3 END       4 Large CMND  5 Log On      6 DataType FC

7 Page bwd   8 Page fwd  9 Page 1    10 Scrl. -->  11 Scrl. <-- 12 Structure
```

Now you can view the other columns in the table.

Note: The Loc field now indicates that the display begins with position 75.

View a Structured Display

To see a structured display of the data table columns, press **PF12**. The structured display lists each column from the Data Table display in left-to-right column order. The following screen shows the structured display of the columns in the previous screen.

```
                        CA InterTest     File DB2 Facility

EXEC SQL                                                    Loc 00001
SELECT * FROM DEVMPL.EMP
------------------------------------------------------------------------------
COL | COLUMN NAME                    | DATATYPE   | LENGTH | CODE | NULL |
----+-------------------------------+------------+--------+------+------+
  1 | EMPNO                         | Char       |      6 |  452 |   No |
  2 | FIRSTNME                      | Char  Var  |     12 |  448 |   No |
  3 | MIDINIT                       | Char       |      1 |  452 |   No |
  4 | LASTNAME                      | Char  Var  |     15 |  448 |   No |
  5 | WORKDEPT                      | Char       |      3 |  453 |  Yes |
  6 | PHONENO                       | Char       |      4 |  453 |  Yes |
  7 | HIREDATE                      | Date Type  |     10 |  385 |  Yes |
  8 | JOB                           | Char       |      8 |  453 |  Yes |
  9 | EDLEVEL                       | Integer S  |      2 |  501 |  Yes |
 10 | SEX                           | Char       |      1 |  453 |  Yes |
 11 | BIRTHDATE                     | Date Type  |     10 |  385 |  Yes |
 12 | SALARY                        | Decimal    |   9. 2 |  485 |  Yes |
 13 | BONUS                         | Decimal    |   9. 2 |  485 |  Yes |
 14 | COMM                          | Decimal    |   9. 2 |  485 |  Yes |
_____________________________________________________________________________
1 Help       2 CMND list 3 END       4 Large CMND  5 Log On     6 DataType FC
7            8           9          10            11           12 Data Table
```

Each row of the Structured Display gives the following information about a column on the Data Table display:

- Relative (left-to-right) column number
- Name
- Data type (such as character and integer)
- Length
- Data type code
- Whether the field contains null characters

Press **PF12** to redisplay the table display.

Select the starting column before redisplaying the Data Table in one of the following ways:

- Change the value in the LOC field to indicate a starting display position. To do this, enter *nn*, clear any remaining LOC field numbers, and press **PF12**.

Example

If you enter 22 in the LOC field, the display starts at position 22.

- Change the value in the LOC field to indicate a starting column number. To do this, enter *#nn*, clear any remaining LOC field numbers, and press **PF12**.

Example

If you enter #9 in the LOC field, the display starts with the first character in column 9.

PF Keys

The PF keys perform the functions shown in the following table when you are using SQL.

PF Key	Function
PF1	Help initiates the CA InterTest for CICS online Help facility.
PF2	CMND List displays a list of the last six commands entered. You can re-execute or modify these commands.
PF3	END or Clear terminates the FILE facility.
PF4	Large CMND extends the command input area.
PF5	Log On/Off toggles between logging on/logging off when the AUDIT option is N.
PF6	Datatype FC changes the DATATYPE to FC (File Control).
PF7	Page Bwd scrolls the current display one page backward. The key setting indicates First Page when you are viewing the first page of the display.
PF8	Page Fwd scrolls the current display one page forward. The key setting indicates Last Page when you are viewing the last page of the display.
PF9	Page 1 returns to the first page of the table data display.
PF10	**==>** scrolls the current display 80 characters to the right.
PF11	**<==** scrolls the current display 80 characters to the left.

PF Key	Function
PF12	Structure toggles between structured column definitions and table data displays.

The legends next to the PF keys show as needed. For example, if the table display contains more rows and columns than are shown on one screen, the legends next to PF7 through PF11 display. If the table display is shown on one screen, the legends for these PF keys do not display.

If you are on the first screen and page backward (PF7), the legend changes from **Page bwd** to **First Page** and is highlighted. Similarly, if you try to page forward past the last row, the legend for PF8 changes from **Page fwd** to a highlighted **Last page**.

FILE Functions for DB2 and SQL/DS Databases

This section shows how to perform the following functions:

- Select columns of data from a table
- Update a column of data in a table
- Delete a row from a table
- Insert a row into a table
- Confirm changes

You also might be able to perform other functions depending on your security authorization.

Select Columns of Data

Use the SELECT command to display one column, many columns, or all columns in a table. The SELECT command also lets you set conditions to determine whether or not a column is displayed.

Example

To view all columns in a table, specify the following, and then press **Enter**:

```
SELECT * FROM DSN8130.TEMPL
```

FILE

Displays all of the data in the table.

Update a Column

The UPDATE command lets you update a column. Set conditions to determine when the update should be performed.

Example

To change the WORKDEPT column for an employee, specify the following, and then press **Enter**:

```
UPDATE DSN8130.TEMPL SET WORKDEPT = `A15' WHERE EMPNO = `000010'
```

This command instructs FILE to change the WORKDEPT column in table DSN8130.TEMPL for the employee whose EMPNO is 000010.

Before performing the update, you are asked to confirm the change. For details, see Confirm Changes (see page 497) later in this chapter.

Delete a Row

Use the DELETE command to delete a row or several rows of data. Set conditions to determine when the deletion should be performed.

Example

To delete a department from a table, specify the following, and then press **Enter**:

```
DELETE FROM DSN8130.TEMPL WHERE WORKDEPT = 'D11'
```

This command instructs FILE to delete from table DSN8130.TEMPL all rows in which the WORKDEPT is D11.

Before performing the deletion, FILE displays a second screen indicating how many rows will be deleted and asking you to confirm the deletion. Enter **C** to confirm the deletion or **R** to cancel it. If you enter C, FILE performs the deletion and informs you that the command was successfully executed.

Insert a Row

The INSERT command lets you insert an entire row of data. First specify the column names, and then specify the values to insert into each column. Set conditions to determine when the insertion should be performed.

- Insert only one row of data at a time.
- You must specify data for each column in the table, except for columns that contain nulls, which are identified by asterisks.

Example

To insert employee data into a table, specify the following, and then press **Enter**:

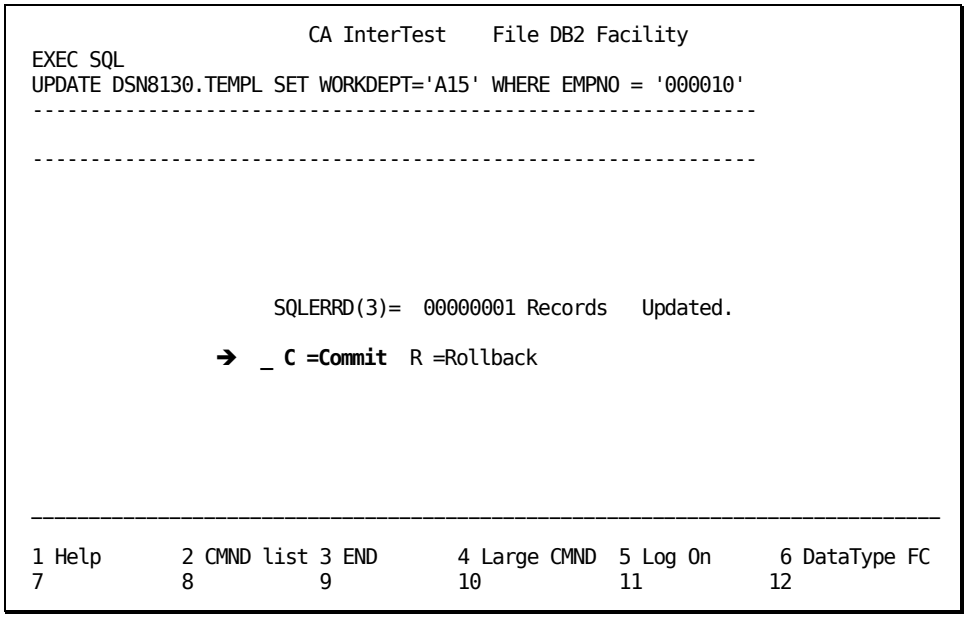
```
INSERT INTO DSN8130.TEMPL
(EMPNO,FIRSTNME,MIDINIT,LASTNAME,WORKDEPT,SALARY)
VALUES(`010200',`JAMES',`S',`NEWMAN',`D11',27500)
```

FILE inserts a row into table DSN8130.TEMPL. This row contains six columns of information for employee James S. Newman.

Before performing the insertion, FILE displays a second screen asking you to confirm or cancel the insertion. Enter **C** to confirm or **R** to cancel the insertion.

Confirm Changes

When you instruct FILE to update a column, or delete or insert a row, FILE displays the following screen:



You must enter either a C or R.

- Enter **C** to confirm that the change is to be executed. FILE executes the change and indicates that the command has been successfully executed.
- Enter **R** to cancel the change.

Redisplay SQL Commands

To redisplay the last six commands you have entered, press **PF2**. Then select any command to re-execute or modify. The following screen shows a command list.

- Enter **x** in the field to the left of a command to execute it
- Enter **m** to modify a command
- If you do not enter any data, any PF key returns you to either the table data display or structured display

```
                    CA InterTest   File DB2 Facility

SQL  Command List          X  to execute     M  to modify the command

_  UPDATE DSN8130.TEMPL SET WORKDEPT='A15' WHERE EMPNO = '000010'


_  DELETE FROM DSN8130.TEMPL WHERE WORKDEPT = 'D11'


_  INSERT INTO DSN8130.TEMPL (EMPNO,FIRSTNME,MIDINIT,LASTNAME,WORKDEPT,SALARY)
   VALUES('010200','JAMES','S','NEWMAN','D11',27500)


_  SELECT * FROM DSN8130.TEMPL
```

View the CICS Attachment Facility (z/OS Users Only)

The Resource Control Table (RCT), or CICS DB2 attachment facility, contains the information required for CICS to establish its connection to DB2. The FILE transaction's DB2 facility lets you view information and statistics on each entry in the active RCT.

The display format is presented as if the RCT existed as a DB2 table. This display is similar to the information provided by the IBM-supplied DSNC DISP STAT command, but provides more complete and easily accessed information.

The following table summarizes the differences between the CA InterTest for CICS DSNCRCT Display and the CICS DSNC Display.

CA InterTest for CICS DSNCRCT Display	CICS DSNC Display
Gives all RCT transactions	Gives only the first transaction in a group
User transactions appear in alphabetical order	Transactions in RCT order
Shows the Authorization ID	Does not show the Authorization ID

To display the RCT, enter the keyword **DSNCRCT** in the DB2 command area. CA InterTest for CICS redisplays the keyword as SELECT * FROM DSNCRCT for compatibility with other DB2 data table displays; however, SELECT command clauses are not supported with DSNCRCT.

The CA InterTest for CICS RCT display includes 15 columns of information, requiring three panels for display. To scroll to the next or previous panel of columns, press **PF11** or **PF10.** To see a Structured Display of the columns, press **PF12**.

The following screen shows the initial RCT display showing columns 1 to 8.

```
                        CA InterTest     File DB2 Facility
EXEC SQL                        # Cols 015   Max Loc 00190   Loc 00001 Page 001
SELECT * FROM DSNCRCT3

1---------2------------3------------4-------5-------6---------7--------8-------

 Tran Id  |  *Plan Id  |  *Auth Id  | Auths | Calls | Commits | Aborts | Waits|
----------+------------+------------+-------+-------+---------+--------+------+

 DSNC     |            |            |     0 |     0 |       0 |      0 |     0|
 POOL     |  DSN8CC21  |            |     0 |     0 |       0 |      0 |     0|
 CNTL     |  FILE21    |  INTRTST   |     0 |     0 |       0 |      0 |     0|
 CORE     |  FILE21    |  INTRTST   |     0 |     0 |       0 |      0 |     0|
 DB2A     |  ASMSQL    |  DEVMPL    |     0 |     0 |       0 |      0 |     0|
 DB2C     |  COBSQL    |  DEVMPL    |     0 |     0 |       0 |      0 |     0|
 DB2P     |  PLISQL    |  DEVMPL    |     0 |     0 |       0 |      0 |     0|
 D8CS     |  DSN8CC21  |            |     0 |     0 |       0 |      0 |     0|
 D8PP     |  DSN8CQ21  |            |     0 |     0 |       0 |      0 |     0|
 D8PS     |  DSN8CP21  |            |     0 |     0 |       0 |      0 |     0|
 D8PT     |  DSN8CH21  |            |     0 |     0 |       0 |      0 |     0|
 D8PU     |  DSN8CH21  |            |     0 |     0 |       0 |      0 |     0|
 FILE     |  FILE21    |  INTRTST   |     0 |     0 |       0 |      0 |     0|
 HELP     |  FILE21    |  INTRTST   |     0 |     0 |       0 |      0 |     0|
_____________________________________________________________________________
1 Help       2 CMND list 3 END       4 Large CMND  5 Log On     6 DataType FC
7 Page bwd   8 Page fwd  9 Page 1   10 Scrl.  ->  11 Scrl. <-   12 Structure
```

The first eight columns are defined next.

Tran ID—Lists all CICS transactions that issue SQL calls. The first two entries are system transactions; the rest are user transactions listed in alphabetical order. The system transactions are defined next.

- DSNC is the command processor transaction; it does not have an associated plan.
- POOL defines threads that are shared by some or all of the CICS transactions. These threads are allocated to transactions only for a CICS unit of work. Consider them short-term threads. Any RCT entry that is specified to overflow to the pool can use them.

Plan ID—Indicates the plan name associated with the transaction.

Auth ID—Indicates the authorization identification concatenated to the table name when the associated transaction is used to access a data table.

Auths—Indicates the number of authorizations (sign-on invocations) for transactions associated with this entry.

Calls—Indicates the number of SQL calls issued by transactions associated with this entry.

Commits—Indicates the number of sync points issued for transactions associated with this entry.

Aborts—Indicates the number of sync point rollbacks and abends issued for transactions associated with this entry.

Waits—Indicates the number of times that all available threads for this entry were busy and the associated transactions had to either wait or be diverted to the pool.

The second panel shows all the DB2 thread statistics, which are kept in the RCT. The following screen shows a sample display of the second panel of columns.

```
                        CA InterTest    File DB2 Facility
EXEC SQL                       # Cols 015   Max Loc 00190   Loc 00080 Page 001
SELECT * FROM DSNCRCT3

9-----------10--------------11-----------12--------------13--------------
| Max Thread | Max Act Thread | High Thread | Curr Thread Lvl | Thread Subtask|
+------------+----------------+-------------+-----------------+---------------+
|          1 |              1 |           0|               0 |              1|
|          3 |              3 |           0|               0 |              0|
|          0 |              0 |           0|               0 |              0|
|          0 |              0 |           0|               0 |              0|
|          2 |              2 |           0|               0 |              0|
|          2 |              2 |           0|               0 |              0|
|          2 |              2 |           0|               0 |              0|
|          1 |              1 |           0|               0 |              0|
|          1 |              1 |           0|               0 |              0|
|          1 |              1 |           0|               0 |              0|
|          1 |              1 |           0|               0 |              0|
|          1 |              1 |           0|               0 |              0|
|          0 |              0 |           0|               0 |              0|
|          0 |              0 |           0|               0 |              0|

_____________________________________________________________________________
1 Help       2 CMND list 3 END       4 Large CMND  5 Log On     6 DataType FC
7 Page bwd   8 Page fwd  9 Page 1   10 Scrl. -->  11 Scrl. <-- 12 Structure
```

Columns 9 through 13 are defined next.

Max Thread—Indicates the maximum number of threads the attachment facility should be prepared to connect for this transaction group.

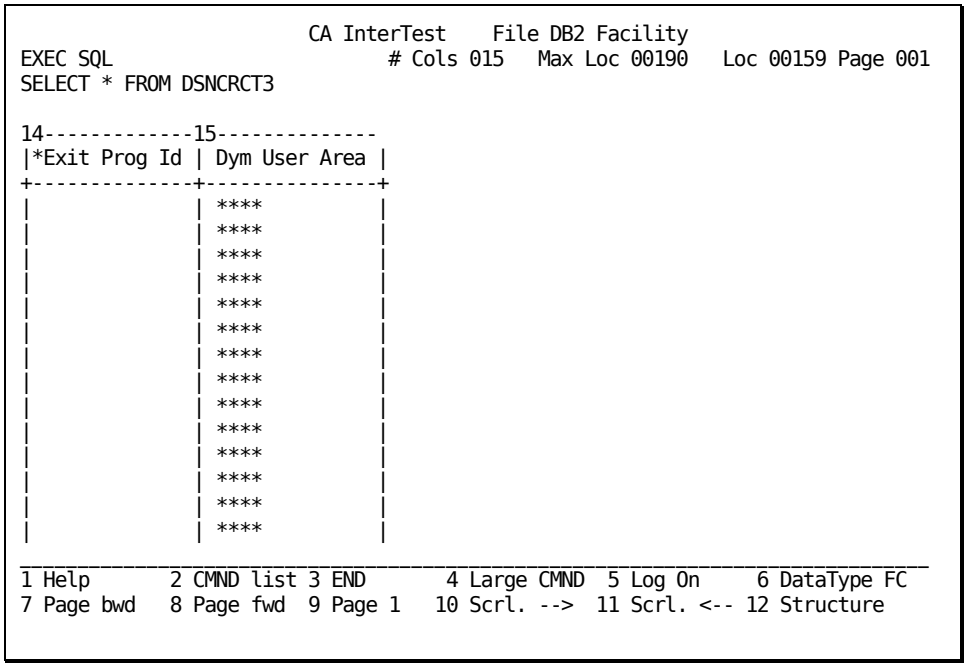
Max Act Thread—Indicates the maximum number of threads the attachment facility should allow to be connected for this transaction, group, or pool before requests must either wait or be diverted to the pool.

High Thread—Indicates the maximum number of threads required by transactions associated with this entry since the connection was started. This number includes transactions forced to wait on a thread or diverted to the pool.

Curr Thread Lvl—Indicates the current number of threads associated with this entry at this time.

Thread Subtask—Indicates the maximum number of z/OS subtasks or threads that should be started when the attachment facility is started.

The third panel contains two columns that deal with dynamic plan selection. The following screen shows columns 14 and 15 of the RCT display.



Exit Prog ID—Indicates the name of the exit program, which dynamically allocates the plan name for transactions associated with this entry.

Dym User Area—Displays the contents of the fullword in the RCT entry reserved for dynamic exit program use.

Note: The RCT statistics are kept on an entry basis, not on a transaction basis. This means that all transactions in a group (sharing the same Plan ID) will have the same statistics.

BDAM Files

The FILE transaction performs any BDAM function allowed in your CICS system.

Note: You might want to consult the *IBM CICS/VS Application Programmer's Reference Manual* for information on using BDAM files. This manual helps you formulate your requests correctly and explains the meaning of many responses you receive from FILE.

To begin processing a BDAM file, first make sure you are viewing the correct screen. From the Auxiliary Storage Menu, select option 1 Files.

The DATATYPE= field should be set to FC. If not, press **PF6** or **Enter**:

```
DATATYPE=FC
```

Then enter:

```
FILEID= file name
```

The other information you enter depends on the function you want to perform.

Note: Undefined record type files are not supported.

FILE Screen Layout for BDAM Files

The following screen shows a blank header area on a FILE screen for BDAM files. For examples of the complete screen, see the Dump Format (see page 447), Character Format (see page 449), Vertical Format (see page 450), Structured Format (see page 451) sections earlier in this chapter.

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

Field	Function
DATATYPE	Identifies the type of data organization. This must be FC.
FILEID	Specifies the name of the BDAM file.
MODE	Identifies the FILE operation, such as browse, update, add, add in mass insertion. This field is blank if you are *viewing* a record.
LOG	Specifies whether FILE's logging facility is on or off.
TODEST	Identifies a transient data destination (when a work area is copied) or a dump identification (when a work area is dumped).
PASSWORD	Must contain a four-character password if the file is password protected.
FUNC	Specifies the FILE function. When this field contains a function code, it is executed if you press Enter. In addition to common FILE functions, enter the following commands: BEGB, DEL, ENDB, GET, GETU, NEXT, PREV, PUT, REL.
SUBFUNC	Modifies a FUNC specification.
RETMETH	Identifies the RCID field (BDAM files only).
ARGTYP	Must contain RBA or XRBA when working with ESDS files.
SRCHTYP	Specifies how RCID information is processed (KSDS files only).
MESSAGE	Contains FILE messages. Also specify messages here when FUNC=COPY or FUNC=DUMP.

Field	Function
RETNRCID	Contains the record key of the retrieved record.
CHGELEN	Specifies the number of bytes changed in the work area when FUNC=CHGE.
RCID	Identifies the record to be processed.
DATA	Contains data needed to perform certain FILE functions.
SIZE	Specifies the record size or the number of records to be searched.
FORMAT	Specifies the format in which the record is displayed.
USE	Specifies the program and structure for structured format.
LOC	Specifies a location within a record.

Some fields are used only for certain tasks. The discussion of each FILE function indicates which fields must be specified.

When a record displays, its FILE information is on the lower-right portion of the screen. This information includes the following:

- DSORG (data organization)
- RECFM (record format)
- LRECL (record length)
- BLKSIZE (blocksize)
- KEYPOS (key position)
- KEYLEN (key length)
- STRNO (string number)

Beneath this information is a list of the functions you can perform, including READ, ADD, UPDATE, BROWSE, DELETE.

FILE Functions for BDAM Records

The following FILE functions are discussed in this section:

- Viewing a BDAM record
- Browsing a BDAM file
- Searching for a data string
- Updating a BDAM record
- Adding a new BDAM record

View a BDAM Record

To view a record, specify the READ operation on the FILE definition.

1. Enter the following information:

   ```
   FUNC=GET
   RCID=record identification
   ```

   and, if appropriate, specify the following *optional* fields:

   ```
   RETMETH=KEY          (Use if RCID contains a deblocking key)
   RETMETH=RELREC       (Use if RCID contains a deblocking relative record number)
   ```

2. Press **Enter**. The following information appears on the FILE screen:

   - The MESSAGE= field shows the following message RECORD OBTAINED FOR VIEWING
   - The MODE= field becomes blank
   - The RETNRCID= field contains the returned record's identification
   - The SIZE= field gives the size of the data in the record, which is the size of the work area
   - The data is displayed, beginning with the byte indicated in the LOC= field

Example

To view a BDAM record that is the sixth record on track 3 in the file BDAMFIL, follow this procedure.

1. Enter:

   ```
   FILEID=BDAMFIL
   FUNC=GET
   RCID=X'000306'
   ```

2. Press **Enter.**

Browse a BDAM File

To browse a file, specify the BROWSE operation on the FILE definition.

1. Press **PF4** or complete the FUNC= field as shown next:

   ```
   FUNC=BEGB
   ```

   and, if appropriate, complete the following *optional* fields:

   ```
   RCID= record identification where browse begins
   RETMETH=KEY       (Use if RCID contains a deblocking key)
   RETMETH=RELREC    (Use if RCID contains a deblocking relative record number)
   ```

2. Press **Enter**. The first record displays and the following information is on the FILE screen:

   - The MESSAGE= field shows the message BROWSE BEGUN
   - The MODE= field indicates BROWSE
   - The FUNC= field contains the function NEXT

3. To view the records in *ascending* sequence, complete the FUNC= field as shown following, and then press **Enter**:

   ```
   FUNC=NEXT
   ```

   The FILE display screen shows the next sequential record.

   Note: You cannot browse a BDAM file in descending sequence.
   To interrupt a browse operation to do something else, such as scroll the display, you must erase the FUNC= field.

4. There are two ways to terminate a browse operation: explicitly and implicitly.

   - To terminate a browse *explicitly*, press **PF4** or specify the following, and press **Enter**:

     ```
     FUNC=ENDB
     ```

     The following information is on the FILE screen:

     - The MESSAGE= field shows the message BROWSE TERMINATED
     - The MODE= field becomes blank

   - To terminate a browse *implicitly*, specify one of the following entries and press **Enter**:

     ```
     FUNC=GET
     FUNC=GETU
     FUNC=ADDN
     FUNC=ADDU
     ```

     The information on the FILE screen reflects the function you selected.

Search for a Data String

This function is valid only for files with the browse capability. Check the lower-right corner of the FILE screen for the properties of the file before attempting to use this function.

1. Complete the following fields as shown next:

   ```
   FUNC= SRCH
   RCID= record id of first record to be searched
   DATA= search argument
   LOC= position in the record to search
   ```

   or

   ```
   LOC=ANY
   ```

   - LOC= *position* instructs FILE to look for the search argument only at the specified position.
   - LOC=ANY instructs FILE to look for the search argument in the entire record.

   In addition, there is an *optional* field:

   ```
   SIZE= number of records to be searched (default=1)
   ```

2. Press **Enter**. If a match is found, the record displays, and the following information is on the FILE screen:
   - The MESSAGE= field shows the message DATA HAS BEEN LOCATED.
   - The RETNRCID= field contains the returned record's identification.
   - The LOC= field contains the location of the found data.
   - If no match is found, the MESSAGE= field shows the message DATA NOT FOUND.

Update a BDAM Record

To update a record, specify the UPDATE operation on the FILE definition.

1. Complete the following fields:

   ```
   FUNC=GETU
   RCID= record identification
   ```

   and, if appropriate, the following *optional* field:

   ```
   RETMETH=KEY       (if RCID contains a deblocking key)
   RETMETH=RELREC    (if RCID contains a deblocking relative record number)
   ```

   Note: You can update any record that was obtained by a GET, BEGB, NEXT, or PREV function. If you already have a record, go to Step 3.

2. Press **Enter**, and the following information appears on the FILE screen:

   - The MESSAGE= field shows the message RECORD OBTAINED FOR UPDATE
   - The MODE= field indicates UPDATE
   - The RETNRCID= field contains the returned record's identification
   - The SIZE= field gives the size of the data in the record (the size of the work area)
   - The data displays, beginning with the byte indicated in the LOC= field

3. Make any necessary changes to the displayed record's data. However, the record will not change until you perform Step 4 and rewrite the updated record. To rewrite the updated record, complete the following fields, and then press **Enter**:

   ```
   FUNC=PUT
   RCID= record identification
   ```

   and, if appropriate, the following *optional* field:

   ```
   SIZE= size of record's data    (required for variable length)
   ```

   The following information appears on the FILE screen:

   - The MESSAGE= field shows the message RECORD UPDATED.
   - The MODE= field becomes blank.
   - The data in the work area continues to display.

4. If you change your mind and decide not to update the record at this time, complete the FUNC= field as shown next, and press **Enter**:

   ```
   FUNC=REL
   ```

   The following information appears on the FILE screen:

   - The MESSAGE= field shows the message FWA HAS BEEN RELEASED**.**
   - The MODE= field becomes blank.
   - The data in the work area continues to display.

Add a New BDAM Record

To add a record, specify the ADD operation on the FILE definition.

Note: Before reading this section, consult the BDAM Data Sets section of the *File Control Section* in the *IBM CICS/VS Application Programmer's Reference Manual*.

1. Complete one of the following fields:

   ```
   FUNC=ADDN       (to obtain a new work area)
   FUNC=ADDU       (to use the data in the existing work area)
   ```

   Complete the following *optional* field, if appropriate:

   ```
   DATA= characters
   ```

   - If you enter FUNC=ADDN, the characters you type in the DATA= field are those that will fill the work area.
   - If you enter FUNC=ADDU, the characters you type in the DATA= field will fill the newly obtained data space (the new work area might be larger than the present one for a variable-length file).

2. Press **Enter**. The following information displays on the FILE screen:
   - The MESSAGE= field shows the message WORK AREA OBTAINED.
   - The MODE= field indicates ADD.
   - The SIZE= field gives the size of the work area (the size is the maximum permitted for this particular file).
   - The obtained work area displays, beginning with the byte indicated in the LOC= field.

3. Make any necessary changes to the newly obtained work area.

4. To write the newly added record, complete the following fields as follows:

   ```
   FUNC=PUT
   RCID= record identification
   ```

   and, if appropriate, the following *optional* field:

   ```
   SIZE= size of record's data  (required for variable length)
   ```

   Press **Enter**. The following information appears on the FILE screen:

   - The MESSAGE= field shows the message RECORD ADDED.
   - The MODE= field becomes blank.
   - The data in the work area continues to display.

5. If you change your mind and decide not to add any records at this time, complete the FUNC= field as shown following, and press **Enter**:

   ```
   FUNC=REL
   ```

   The following information appears on the FILE screen:

   - The MESSAGE= field shows the message FWA HAS BEEN RELEASED.
   - The MODE= field becomes blank.
   - The data in the work area continues to display.

Temporary Storage

The FILE transaction performs any temporary storage function allowed in your CICS system.

To begin processing temporary storage, first make sure you are viewing the correct screen. From the Auxiliary Storage Menu, select option 5 TS Queues. The DATATYPE= field should be set for TS. If not, press PF6 or enter:

```
DATATYPE=TS
```

The other information you enter depends on the function you want to perform.

FILE Screen Layout for Temporary Storage

The following screen shows a blank header area on a FILE screen for temporary storage. For examples of the complete screen, see Dump Format (see page 447), Character Format (see page 449), Vertical Format (see page 450) and Structured Format (see page 451) earlier in this chapter.

```
DATATYPE= TS                        LOG=OFF     TODEST=
FUNC=      SUBFUNC=      STORFAC=
MESSAGE=

    RCID=                                                        CHGELEN=

    DATA=                                                         SIZE=0000

FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*

LOC 0000  ........ ........ ........ ........  ................




--------------------------------------------------------------------------------

1 Help       2 Format C   3 End        4            5            6 DataType FC
7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

The fields provide the following information:

Field	Function
DATATYPE	Identifies the type of data organization. This must be TS.
LOG	Specifies whether FILE's logging facility is on or off.
TODEST	Identifies a transient data destination (when a work area is copied) or a dump identification (when a work area is dumped).
FUNC	Specifies the FILE function. When this field contains a function code, it is executed if you press Enter**.** In addition to common FILE functions, enter the following commands: GET, GETQ, PURG, PUT, PUTQ, REL.
SUBFUNC	Modifies a FUNC specification.
STORFAC	When writing a record (FUNC=PUT or PUTQ), specifies the type of storage: AUX (auxiliary) or MAIN.
ENTRY	Specifies the relative position of a record in a temporary storage queue. Required for FUNC=GETQ, PUTQ, and SRCH.
MESSAGE	Contains FILE messages. Also specify messages here when FUNC=COPY or FUNC=DUMP.

Field	Function
CHGELEN	Specifies the number of bytes changed in the work area when FUNC=CHGE.
RCID	Identifies the record or queue to be processed.
DATA	Contains data needed to perform certain FILE functions.
SIZE	Specifies the record size or the number of records to be searched.
FORMAT	Specifies the format in which the record is displayed.
USE	Specifies the program and structure for structured format.
LOC	Specifies a location within a record.

Some fields are used only for certain tasks. The discussion of each FILE function indicates which fields must be specified.

FILE Functions for Temporary Storage

The following FILE functions are discussed in this section:

- Viewing a temporary storage queue record
- Adding a new temporary storage queue record
- Purging a temporary storage queue
- Searching a temporary storage queue for a data string

Viewing a Temporary Storage Queue Record

Follow this procedure to view a temporary storage queue record:

1. Complete the following fields:

```
FUNC=GETQ
RCID= temporary storage queue identification
ENTRY= record's relative position in the queue  (default=1)
```

2. Press Enter. The record's data is displayed on the FILE screen with the following information:
   - The MESSAGE= field shows the message TS QUEUE RECORD RETRIEVED.
   - The SIZE= field gives the size of the retrieved record's data.

Add a New Temporary Storage Queue Record

Follow this procedure to add a temporary storage queue record:

1. Specify one of the following:

   ```
   FUNC=ADDN    (to obtain a new work area)
   FUNC=ADDU    (to use the data in the existing work area)
   ```

   Then complete the SIZE= field as shown next:

   ```
   SIZE= size of data area
   ```

   In addition, complete the following *optional* field, if appropriate:

   ```
   DATA= characters
   ```

   - If you enter FUNC=ADDN, the characters you type in the DATA= field will fill the work area.
   - If you enter FUNC=ADDU, the characters you type in the DATA= field will fill the newly obtained data space (if the new work area is larger than the present one).

2. Press **Enter**. The following information displays on the FILE screen:
   - The MESSAGE= field shows the message WORK AREA OBTAINED.
   - The SIZE= field gives the size of the work area (the size will be the maximum permitted).
   - The obtained work area is displayed, beginning with the byte indicated in the LOC= field.
3. Make any necessary changes to the newly obtained work area.

4. To write the newly added record, complete the following fields:

```
FUNC=PUTQ
ENTRY= record's relative position in queue
RCID= temporary storage identification
SIZE= size of data in the record
```

and, if appropriate, the following two *optional* fields:

```
SUBFUNC= REPL       (to replace an existing record)
STORFAC=AUX
```

or

```
STORFAC=MAIN
```

- Use STORFAC=AUX to place the record on a direct access storage device. This is the default.
- Use STORFAC=MAIN to place the record in main storage.

5. Press **Enter**. The following information displays on the FILE screen:
   - Without SUBFUNC=REPL, the MESSAGE= field shows the message TS QUEUE RECORD WRITTEN.
   - With SUBFUNC=REPL, the MESSAGE= field shows the message TS QUEUE RECORD WRITTEN AND REPLACED.
   - The SIZE= field gives the size of the work area.
   - The work area displays, beginning with the byte indicated in the LOC=field.

Examples

You want to add a new 100-byte record to a temporary storage queue called TEMPSTO1, which resides in auxiliary storage. This record is to be the last record in the queue.

Specify the following, and then press **Enter**:

```
FUNC=ADDN
RCID=C'TEMPSTO1'
SIZE= 100 or 64         (for FORMAT=D)
```

Make changes to the work area.

Specify the following, and then press **Enter**:

```
FUNC=PUTQ
RCID=C'TEMPSTO1'
```

To replace record 5 in a temporary storage queue named TEMPSTO2, which resides in main storage, using the work area from Example 1:

Specify the following, and then press **Enter**:

```
FUNC=ADDU
```

Make changes to the work area.

Specify the following, and then press **Enter**:

```
FUNC=PUTQ
ENTRY=05
RCID=C'TEMPSTO2'
SUBFUNC=REPL
SIZE= 100 or 64          (for FORMAT=D)
STORFAC=MAIN
```

Purge a Temporary Storage Queue

Follow this procedure to purge a temporary storage queue:

1. Complete the following fields:

   ```
   FUNC=PURG
   RCID= temporary storage queue identification
   ```

2. Press **Enter**. The following information is on the FILE screen:
   - The MESSAGE= field shows the message *xxx* RECORDS DELETED.
   - The SIZE= field gives the size of the work area.
   - The work area displays, beginning with the byte indicated in the LOC= field.

Search a Temporary Storage Queue for a Data String

Follow this procedure to search a temporary storage queue for a data string.

1. Complete the following fields:

   ```
   FUNC=SRCH
   ENTRY= relative record position to start search
   RCID= temporary storage queue identification
   DATA= search argument
   LOC= position in record to search for the data string
   ```

   or

   ```
   LOC=ANY     (to search the entire record for the data string)
   ```

   In addition, complete the following *optional* field:

   ```
   SIZE= number of records to search           (default=1)
   ```

2. Press **Enter**. If a match is found, the record containing the data displays, starting at the location where the data was found. The following information also appears on the FILE screen:

   - The MESSAGE= field shows the message DATA HAS BEEN LOCATED.
   - If no match is found, the MESSAGE= field shows the message DATA NOT FOUND.
   - The ENTRY= field contains the returned record's relative position in the queue.
   - The LOC= field contains the location of the found data.

Example

To scan ten records in the temporary storage queue named TEMPSTO1 starting at the third record for the data string TEST, if that data is in location 20 in the record, specify the following, and then press **Enter**:

```
FUNC=SRCH
ENTRY=03
RCID=C'TEMPSTO1'
DATA=C'TEST'
LOC=00020 OR 0013              (for FORMAT=D)
SIZE=0010 OR 000A              (for FORMAT=D)
```

Transient Data

The FILE transaction performs any transient data function allowed in your CICS system.

To begin processing transient data, first make sure you are viewing the correct screen. From the Auxiliary Storage Menu, select option 5 TD Queues. The DATATYPE= field should be set at TD. If not, press **PF6** or **Enter**:

```
DATATYPE=TD
```

The other information you enter depends on the function you want to perform.

FILE Screen Layout for Transient Data

The next screen shows a blank header area on a FILE screen for temporary storage. For examples of the complete screen, see Dump Format (see page 447), Character Format (see page 449), Vertical Format (see page 450) and Structured Format (see page 451) earlier in this chapter.

```
DATATYPE= TD                        LOG=OFF     TODEST=
FUNC=      SUBFUNC=


MESSAGE=                                                        CHGELEN=
    DATA=                                                        SIZE=0000
FORMAT= D 00112233 44556677 8899AABB CCDDEEFF  *0123456789ABCDEF*
LOC 0000  ........ ........ ........ ........  ................









--------------------------------------------------------------------------------
1 Help       2 Format C   3 End        4            5            6 DataType FC
7 Page bwd   8 Page fwd   9 Caps Off  10 Top       11 Bottom    12
```

The fields provide the following information:

Field	Function
DATATYPE	Identifies the type of data organization. This must be TD.
DESTID	Specifies the name of the transient data, as defined in the DCT.
LOG	Specifies whether FILE's logging facility is on or off.
TODEST	Identifies a transient data destination (when a work area is copied) or a dump identification (when a work area is dumped).

Field	Function
FUNC	Specifies the FILE function. When this field contains a function code, it is executed if you press Enter. In addition to common FILE functions, enter the following commands: GET, PUT.
SUBFUNC	Modifies a FUNC specification.
MESSAGE	Contains FILE messages. Also specify messages here when FUNC=COPY or FUNC=DUMP.
CHGELEN	Specifies the number of bytes changed in the work area when FUNC=CHGE.
RCID	Identifies the record or queue to be processed.
DATA	Contains data needed to perform certain FILE functions.
SIZE	Specifies the record size or the number of records to be searched.
FORMAT	Specifies the format in which the record is displayed.
USE	Specifies the program and structure for structured format.
LOC	Specifies a location within a record.

Some fields are used only for certain tasks. The discussion of each FILE function indicates which fields must be specified.

FILE Functions for Transient Data Records

The following FILE functions are discussed in this section:

- Retrieving a transient data record
- Adding a new transient data record
- Purging intrapartition data

Note: If you are running CTS Version 1.3 and higher, 16-byte and 32-hexadecimal character TS queue names are supported.

Retrieve a Transient Data Record

Note: When you retrieve a record from an intrapartition transient data destination, the retrieved record disappears from that queue.

1. To retrieve a transient data record, complete the following fields:

   ```
   FUNC=GET
   DESTID= transient data destination
   ```

   You are advised to enter the DESTID= field before entering the FUNC= field when you begin working with a particular destination. If only the DESTID= is entered, the properties of the transient data queue are shown for your verification. This can, for example, help you avoid entering a FUNC=GET for an extrapartition destination that is now open for output.

2. Press **Enter**. The record you are retrieving opens on the FILE screen. The following information also displays:

   - The MESSAGE= field shows the message TD RECORD OBTAINED.
   - The SIZE= field gives the length of the data.

Add a New Transient Data Record

Follow this procedure to add a new transient data record:

1. Complete the FUNC= and DESTID= fields, and then complete the SIZE= field:

   ```
   DESTID= transient data destination
   FUNC=ADDN     (to obtain a new work area)
   ```

   or

   ```
   FUNC=ADDU     (to use the data in the existing work area)
   SIZE= size of the data area Note: The SIZE cannot be higher than permitted for
   the destination.
   ```

   In addition, you can complete the following *optional* field if appropriate:

   ```
   DATA= characters
   ```

   - If you enter FUNC=ADDN, the characters you type in the DATA= field will fill the work area.
   - If you enter FUNC=ADDU, the characters you type in the DATA= field will fill the newly obtained data space (if the new work area is larger than the present one).

   You are advised to enter the DESTID= field before entering the FUNC= field when you begin working with a particular destination. If only the DESTID= is entered, the properties of the transient data queue are shown for your verification. This can, for example, help you avoid entering a FUNC=ADDU for an extrapartition destination that is now open for input.

2. Press **Enter**. The following information is on the FILE screen:
   - The MESSAGE= field shows the message WORK AREA OBTAINED.
   - The SIZE= field gives the size of the work area (the size is the maximum permitted).

   The obtained work area is displayed, beginning with the byte indicated in the LOC= field.
3. Make any necessary changes to the newly obtained work area. To write the newly added record, complete the following fields:

   ```
   FUNC=PUT
   DESTID= transient data destination
   ```

   and, if appropriate, the following *optional* field:

   ```
   SIZE= size of the record         (if smaller than work area)
   ```
4. Press **Enter**.

   The MESSAGE= field on the FILE screen shows the message TD RECORD WRITTEN.

Examples

To add a new 100-byte record (with most of its fields blank) to the transient data destination CSSL:

Specify the following, and then press **Enter**:

```
FUNC=ADDN
DESTID=CSSL
DATA=C' '
SIZE= 100   or   64     (for FORMAT=D)
```

Make changes to the work area.

Specify the following, and then press **Enter**:

```
FUNC=PUT
DESTID=CSSL
```

To add another new record to the transient data destination in Example 1, using the work area from Example 1:

Specify the following, and then press **Enter**:

```
FUNC=ADDU
```

Make changes to the work area.

Specify the following, and then press **Enter**:

```
FUNC=PUT
DESTID=CSSL
```

Purge Intrapartition Data

Follow this procedure to purge intrapartition data:

1. Complete the following fields:

   ```
   FUNC=PURG
   DESTID= transient data destination
   ```

   You are advised to enter the DESTID= field before entering the FUNC= field when you begin working with a particular destination. If only the DESTID= is entered, the properties of the transient data queue are shown for your verification. This can, for example, help you avoid entering a FUNC=PURG for an extrapartition destination.

2. Press **Enter**. This purges the data. The MESSAGE= field on the FILE screen displays the message DESTINATION PURGED.

Chapter 12: Using CA InterTest for CICS with DB2 and SQL/DS Programs

This chapter explains how to use CA InterTest for CICS to test CICS/DB2 and SQL/DS application programs, and is intended primarily for system programmers and database administrators.

Preliminary Steps

Before you can monitor a DB2 or SQL/DS program, make sure that the following tasks have been completed:

1. CA InterTest for CICS was installed with DB2 or SQL/DS support.
2. The application program was compiled or assembled with symbolic support.
3. The CICS/DB2 attach facility (z/OS) or SQL/DS Resource Adapter is active.

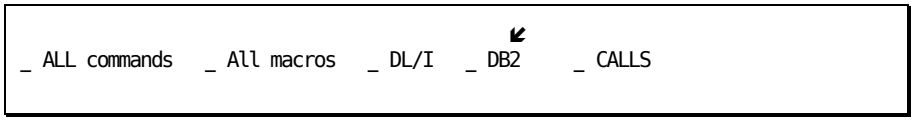
Check for DB2 or SQL/DS Support

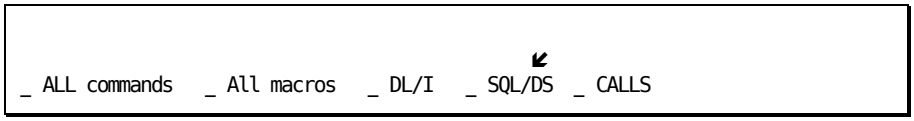
CA InterTest for CICS should have been installed with DB2 or SQL/DS support if you are going to monitor DB2 or SQL/DS application programs.

To ensure that DB2 or SQL/DS support was installed, follow these steps:

1. Access the Request Breakpoint Selection menu. This menu is displayed when you select the Request option from a Monitoring program, such as Monitoring Programs (2.1), Monitoring Transactions (2.2), or Monitoring Terminals (2.3). The chapter "Using Breakpoints" describes how to access this menu.

2. Inspect the first input line on the menu. If it contains either of the following, DB2 or SQL/DS support was installed. If the DB2 or SQL/DS field is missing, support is not installed.DB2 or SQL/DS support was installed. If the DB2 or SQL/DS field is missing, support is not installed.





If you monitor DB2 or SQL/DS programs without the appropriate CA InterTest for CICS support, your programs are halted at an automatic breakpoint each time the program issues a DB2 or SQL/DS call. If necessary, contact your system programmer and request DB2 or SQL/DS support.

Note: The *Installation Guide* explains how to install this support.

Compile or Assemble Programs with Symbolic Support

To test DB2 and SQL/DS programs thoroughly, they should be compiled or assembled with symbolic support, and the program's Cross-Reference should be saved in the CA InterTest for CICS Symbolic File.

For more information:

Breakpoint Displays for NonSymbolic Programs (see page 545)

Check That the Attach Facility or Resource Adapter Is Active

For z/OS Users You must ensure that the CICS/DB2 Attach facility is active in the CICS region in which you are testing DB2 programs.

To test whether the attach facility is active, follow these steps:

1. Type FILE on a clear screen.
2. Change the DATATYPE to DB and press **Enter**.

If the attach facility is active, CA InterTest for CICS displays the following message:

CAIN1701

ENTER THE SQL COMMAND TO BE EXECUTED.

If the attach facility is **not** active, CA InterTest for CICS displays the following message:

CAIN1712

DSNC NOT ACTIVE OR DB2 NOT AVAILABLE.

If the Resource Adapter is active, it displays the following message:

ARI0401I

SQL/DS ONLINE RESOURCE ADAPTER IS ALREADY ACTIVE IN THIS PARTITION.

If the Resource Adapter is **not** active when SQL/DS is in multipartition mode, the following messages display:

ARI0418A

SQL/DS IS NOT READY. RETRY THE ENABLE TRANSACTION CIRB AFTER SQL/DS STARTS.

ARI0413I

RESOURCE ADAPTER ARI0OLRM IS DISABLED.

If the Resource Adapter is active in the non-CICS partition but **not** yet active in the CICS partition, the following messages display:

ARI407E

ERROR ATTEMPTING TO EXEC SQL.

ARI407E

SQLCODE = -560 SQLERRD1 = -180 SQLERRD2 = 0.

ARI413I

RESOURCE MANAGER ARI0OLRM IS DISABLED.

If the Attach Facility or Resource Adapter Is Not Active

If the attach facility or Resource Adapter is **not** active and you execute a DB2 or SQL/DS program, the transaction fails with an AEY9 abend code. If you are monitoring the program with CA InterTest for CICS and you do not have a Handle Condition for this error, your program is halted at an automatic breakpoint at a call to DB2 (CALL 'DSHNLI') or SQL/DS (CALL 'ARIPRDI'). In this case, follow these steps:

1. Call your system programmer or console operator and request that the CICS/DB2 attach facility or SQL/DS Resource Adapter be started.
2. Continue program execution by re-executing the call. In COBOL, this means paging back to the statement prior to the call:

   ```
   PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
   ```

3. Type **G** to the left of this statement and press **PF5**.

Halt Programs Before and After Each SQL Request

To test DB2 and SQL/DS programs, halt the program before and after each SQL request.

- Halting the program **before** each SQL request lets you check that all fields in the request have been initialized correctly
- Halting the program **after** each SQL request lets you view the data returned to your program and the SQL return codes

Halt a Program Before Each SQL Request

To halt a program **before** each SQL request, use the CA InterTest for CICS request breakpoint capability:

1. Access the Request Breakpoint Selection menu, as explained in the chapter "Using Breakpoints."
2. Type **X** in the DB2 or SQL/DS field at the top of the menu. Press **Enter**.

This instructs CA InterTest for CICS to set a breakpoint prior to each **application** call to DB2 or SQL/DS.

- For DB2, this occurs at the call to DSNHLI.
- For SQL/DS, this occurs at the call to ARIPRDI.

When the program halts at the request breakpoint, you can inspect the values of fields. For details, see Inspect Host Variables (see page 527) later in this chapter.

Halt a Program after an SQL Request

When a program halts at a request breakpoint, you can execute the SQL request and stop the program again.

- For COBOL or PL/I programs, press **PF10**, which executes one COBOL verb or one PL/I statement
- For Assembler programs, press **PF10** twice to execute two instructions

When the program is halted after the call, you can inspect the values of fields again and analyze the SQL return codes.

Inspect Host Variables

When you are stopped at a breakpoint, inspect the values of data fields. SQL terminology refers to the fields used in SQL requests as *host* variables. Host variables are used to do the following tasks:

- Supply a value to an SQL request
- Receive a value from an SQL request

A typical SQL request has the following format:

```
EXEC SQL
        SELECT   column names
        INTO     host variables
        FROM     table name
        WHERE    column names/host variables
END-EXEC
```

Example:

- The INTO clause of the SELECT statement specifies host variables.
- The WHERE clause of the SELECT statement uses host variables to qualify the request for data.

In a typical COBOL program:

- All the host variables used to receive data for column values named in an INTO clause are grouped together in a single structure designated by a 01-level field name
- All the host variables used to qualify the data request in a WHERE clause are grouped together in a single structure
- All the host variables used for null indicators are grouped together in a single structure

Notes:

- When your program is halted at a breakpoint **before** an SQL request, check the null indicators and host variables used in the WHERE clause
- When your program is halted **after** an SQL request, check the host variables specified in the INTO clause

Display Host Variables

You can view host variables either individually by name or as a single structure. The following screen shows how CA InterTest for CICS displays host variables. The chapter "Breakpoint Activities" explains how to display the values of host variables.

```
            CA InterTest  MAIN STORAGE UTILITY  Termid = LXD6
Starting at Address =  140D9C               Hexadecimal            Character

01 DCLEMP                          |                              |
 02 EMPNO                          |  F0F0F0F0 F1F0               | 000010
 02 FIRSTNME                       |                              |
  03 FIRSTNMELEN                   |  0009                        | ..
  03 FIRSTNMETEXT                  |  C3C8D9C9 E2E3C9D5 C5000000 | CHRISTINE..
 02 MIDINIT                        |  C1                          | A
 02 LASTNAME                       |                              |
  03 LASTNAMELEN                   |  0004                        | ..
  03 LASTNAMETEXT                  |  C8C1C1E2 00000000 00000000 | HAAS.......
                                   |  000000                      | ...
 02 WORKDEPT                       |  C1F0F0                      | A00
 02 PHONENO                        |  F3F9F7F8                    | 3978
 02 HIREDATE                       |  F1F9F6F5 60F0F160 F0F1      | 19650101
 02 JOB                            |  D7D9C5E2 40404040           | PRES
 02 EDLEVEL                        |  0012                        | ..
 02 SEX                            |  C6                          | F

PF1 Help       2            3 End        4 Return     5            6
PF7 Backward   8 Forward    9 Caps Off  10           11 Redisplay 12 Structure
CORE='DCLEMP'
```

For DB2 Users

If you use the DCLGEN function to generate DB2 table declarations, the COBOL 01-level field name is a concatenation of the characters DCL and the table name, as shown in the previous example.

SQL Return Codes

The SQL communication area (SQLCA) is a structure of variables that is updated after each SQL statement executes. An application program that contains executable SQL statements must provide an SQLCA to let DB2 or SQL/DS to communicate with the program.

Note: The IBM manual *DB2 Application Programmer Guide* provides information on the SQLCA.

When DB2 or SQL/DS processes an SQL statement, it places a return code in the SQLCODE field of the SQLCA that indicates the success or failure of the SQL request:

- If the return code is **zero**, the SQL request was processed. DB2 or SQL/DS did not detect an error or an exceptional condition.
- If the return code is a **negative** number, DB2 or SQL/DS detected an error.
- If the return code is a **positive** number, DB2 or SQL/DS detected an exceptional but valid condition.

Note: Sometimes a zero return code does not indicate a successful execution. You should always check the SQLWARN field to ensure that it is blank, which indicates successful execution. If the field is not blank, a warning condition exists. Check the other warning indicators to determine the specific condition.

Example

If a value was truncated the SQLCODE will be zero, but the SQLWARN1 field indicates truncation.

Because the SQLCA contains valuable diagnostic information, it is a good idea to check its contents after each SQL request.

How to Handle SQL Error Codes

Two techniques are commonly used by programmers to handle SQL error return codes after each SQL request:

- Check the SQLCODE data field immediately after each SQL request
- Branch to a common error handling routine

Checking the SQLCODE immediately tells you which request produced the error so you can create an error message reflecting that request. However, this method might not be efficient if you have many similar routines throughout your program—one for each SQL request. Moreover, if you discover a logic error in such an error handling routine, you must correct the error in all the routines throughout the program.

Branching to a common error handling routine results in a smaller program and simplifies the task of debugging the routine. However, you might not be able to determine which SQL request caused the error unless you pass an indicator to the error routine, which is then reflected in the generated error message.

Users who want to check the SQLCODE field in the SQLCA after each SQL request should set unconditional breakpoints at each occurrence of SQLCODE. The chapter "Using Breakpoints" explains how to set breakpoints at each occurrence of a data field. Your program must be compiled or assembled symbolically with the option to produce a Cross-Reference listing.

Users who branch to a common error handling routine should set request breakpoints at **all** DB2 or SQL/DS calls and then single-step through the breakpoint, as described in the section Halt Programs Before and After Each SQL Request. This technique does not require that programs be compiled or assembled symbolically.

In either case, keep SQLCODE in the Keep window so its value can be checked easily. The Keep window is described in the chapter "Breakpoint Activities."

Display the SQLCODE

Once your program halts at a breakpoint, the Keep window displays the hexadecimal contents of SQLCODE. To display the SQL return code in decimal format, follow these steps:

1. Type **D** to the left of SQLCODE in the Keep window.
2. Press **Enter**.

If you have not kept SQLCODE in the Keep window, follow these steps:

1. Type **D** to the left of the statement where you are stopped.
2. Position the cursor under the data field **SQLCODE**.
3. Press **Enter**.

CA InterTest for CICS displays a CORE structured display of SQLCODE and the following message:

```
CORE053 VALUE OF FIELD is = +/- nnn
```

nnn

Indicates the SQL return code.

DB2 and SQL/DS return codes can have many meanings and subcodes. The *DB2 Messages and Codes* manual explains the return codes.

z/OS users can convert this value into a text message by overkeying the CORE='SQLCODE' command at the bottom of the SQLCODE display with the following command and pressing Enter:

```
CORE=SQLRCODE
```

z/OS users can access the SQLCODE text message display from the breakpoint screen by entering **CORE=SQLRCODE** on the command line, if the command line displays.

CA InterTest for CICS converts the decimal value to a text message. In addition, if there are any warning codes in the SQLCA, CA InterTest for CICS displays each of them along with its meaning. The following is a sample SQLRCODE display.

```
CA InterTest  MAIN STORAGE UTILITY  Termid = LXD6

 DSNT408I SQLCODE = 204

          ERROR:  ADM.DEPT IS AN UNDEFINED NAME









 ------------------------------------------------------------------------

 PFKEYS 1 Help     2          3 End       4 Return      5           6
        7          8          9          10            11          12
  CORE=SQLRCODE
```

DB2 messages often contain tokens that are replaced at execution time. The message in the previous example the token name is replaced by the actual name that is undefined—in this case, ADM.DEPT.

Without CA InterTest for CICS, tokens are often hard to interpret. They are returned in the SQLERRM field of the SQLCA, and if there are multiple tokens, appear sequentially in the order in which they appear in the message text, separated by X'FF'. CA InterTest for CICS inserts tokens in the correct places in the message text, as shown in the previous figure, which makes the DB2 messages much easier to understand.

Display All the Fields in the SQLCA

To view all the fields in the SQLCA after an SQL request, overtype the CORE command at the bottom of the display with one of the following, and press **PF12**:

```
CORE='SQLCA'            (COBOL and PL/I)
CORE='Rn.SQLCA'         (Assembler)
```

n

Indicates the register used to reference the SQLCA.

CA InterTest for CICS displays all the fields in the SQLCA in structured format as defined in the program.

For z/OS, another method for viewing the SQLCA fields is to type the following command and press **PF12**:

```
CORE=SQLCA
```

CA InterTest for CICS displays the SQLCA fields using Assembler language field names.

Both methods should produce the same results, but the second method is available for programs not compiled or assembled symbolically.

Note: Neither command is valid until after an SQL request is issued.

Advanced Debugging Techniques

For specialized debugging tasks, you can instruct CA InterTest for CICS to halt your program at DB2 or SQL/DS calls other than an SQL application request. However, halting programs at these calls usually does not produce much useful information and might create problems.

To halt a program **before** selected DB2 or SQL/DS calls, use the CA InterTest for CICS request breakpoint capability, and follow these steps:

1. Access the Request Breakpoint Selection menu, as explained in the chapter "Using Breakpoints."
2. Enter **X** to the left of the CALLs field at the top of the menu. Press **Enter**.

CA InterTest for CICS displays the Request Breakpoint Selection menu for calls.

```
      CA InterTest MONITORING COMMAND BUILDER  REQUEST BREAKPOINT SELECTION

                     Set  one or more types of CALLs in:
                               PROG=COBSQL
      _ All CALLs

      _ DSNHLI       _ DSNHADDR     _ DSNTIAR      _ ILBOINS0     _ COMDSTUB
      _ TESTCONV




                Enter 'n' to stop only every n'th time   ____


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

For DB2 this menu contains three relevant entries, as shown in the previous panel:

`DSNHLI, DSNHADDR, and DSNTIAR.`

DSNHLI—The call generated for all SQL application requests. To halt the program at these calls, set request breakpoints at all DB2 calls on the first Request Breakpoint Selection menu as described in Halt Programs Before and After Each SQL Request (see page 526) earlier in this chapter.

DSNHADDR—The call made to the SQL-INITIAL routine that is inserted by the DB2 preprocessor in the COBOL Procedure Division to initialize the parameter lists needed for an SQL call.

The DB2 preprocessor generates structures identified at the 01-level by SQL-PLIST*n* in COBOL Working-storage. There is one structure for each EXEC SQL command in the program. The SQL-PLIST*n*s are parameter lists passed to DB2 at each SQL request. The values in the parameter lists are updated at application program initialization using a DSNHADDR call. The SQL-INITIAL routine makes one call to DSNHADDR for each EXEC SQL command. Usually, setting a breakpoint at each DSNHADDR call will not produce much useful information.

DSNTIAR—The call made to the IBM-provided DB2 subroutine that converts an SQL return code to a text message. The subroutine is approximately 32 KB in size and must be statically linked with your program. According to IBM, calling DSNTIAR dynamically from a CICS application program causes problems because of the GETMAINs issued by DSNTIAR.

Unless your application program needs the SQL return code text message, there is no need to set breakpoints at DSNTIAR calls. Instead, use the SQLRCODE command explained in the Handle SQL Return Codes section earlier in this chapter.

DSNC Abends (z/OS Only)

A DSNC abend means the CICS attach facility detected an internal error and has abnormally terminated a DB2 internal task or a user connected task. The CICS attach facility first writes a SNAP dump to the SYS1.DUMP*nn* data set. Then, depending on the type of error, the CICS attach facility either:

- Terminates the user transaction by issuing an EXEC CICS ABEND with DSNC as the abend code. In this case, the application's SQLCA remains set to low values.
- Lets the transaction continue, but produces a dump by issuing EXEC CICS DUMP with DSNC as the abend code. In this case, a return code is placed in the application's SQLCA.

In the first case described previously, you cannot handle the DSNC abend in your program's EXEC HANDLE ABEND routine because control does not come back to it. EDF captures the DSNC abend, but its halt screen provides little useful information.

CA InterTest for CICS intercepts a DSNC transaction abend and produces an automatic breakpoint, as follows.

```
                CA INTERTEST  PROTDEM FILE SOURCE LISTING BREAKPOINT
Command ===>
Program= COBSQL   Option #       Stmt #                            Margin= 01
                                       Search=


   01414*****ENDEXEC.
   01415     PERFORM SQLINITIAL UNTIL SQLINITDONE
 A   ==>     CALL 'DSNHLI' USING SQLPLIST6.
     ==>
     ==> CICS abend intercepted. Abend code = DSNC, after an external CALL.
     ==>
     ==>      Press PF1 for a detailed description.
     ==>
   01417
   01418
_  01419     IF SQLCODE = 530
_  01420        MOVE 'INVALID WORK DEPARTMENT ' TO ERRMSGO
_  01421        MOVE 1                          TO WDEPTL
_  01422        MOVE DFHBMBRY                   TO WDEPTA
_  01423        GO TO SENDRTN.
   01424
```

If CICS terminated the transaction, you cannot continue processing it because the registers were destroyed.

CA InterTest for CICS provides several CORE commands to help you determine the cause of the problem online without waiting for the DB2 SNAP dump.

The CICS Life of Task Control Block (CLOT)

According to the *DB2 Problem Determination Guide*, first locate the CICS Life of Task (CLOT) control block. The reason for the error can be determined by the codes in the CLOTCFLG field of the control block included in the dump.

To display the CLOT, type the following command and press **PF12**:

CORE=CLOT

Next, the *DB2 Problem Determination Guide* recommends that you inspect the CLOTCFLG field. You can page forward to display this field or type the following command and press **PF12**:

CORE=CLOT.CLOTCFLG

Reason and Abend Codes--Inspecting and Interpreting

If the thread subtask abended (CLOTCFLG=18), the reason code is contained in the CLOTWRK1 field and the abend code is contained in the CLOTWRK2 field. Both abend reason codes and subsystem termination reason codes have the following format:

X'00ccxxxx'

X'00'

Specifies the high-order byte

X'cc'

Specifies the hexadecimal sub-component identifier

X'xxxx'

Specifies the hexadecimal reason code

To display this information on a single diagnostic screen, type the following command and press **Enter**:

CORE=SQLRCODE

A sample diagnostic CA InterTest for CICS display follows.

```
              CA InterTest  MAIN STORAGE UTILITY  Termid = LXD6


Transaction DB2C, Program COBSQL   has terminated with an DSNC abend.

         The reason code for this failure is:          18
             The thread subtask has abended; see subtask dump.


         The failing DB2 subcomponent identifier is: 00E7
             Relational data system.

         The subsystem termination reason code is:   000D

Refer to the IBM DATABASE 2 Messages and Codes, Part 4. Reason Codes,
for detailed information.



--------------------------------------------------------------------------------
PFKEYS 1 Help     2           3 End       4 Return      5            6
       7          8           9          10            11           12
 CORE=SQLRCODE
```

Note: For an explanation of the codes, see the *DB2 Messages and Codes* manual.

Inspect the Application Invocation Parameter List (RDIIN)

In some cases, the *DB2 Problem Determination Guide* suggests that you inspect the Application Invocation Parameter List (RDIIN) in the DB2 SNAP dump for further information. If this control block exists in your CICS DSNC dump, you can view it online by typing the following command and pressing **PF12**:

```
CORE=RDIIN
```

View the CICS DB2 Attachment Facility (z/OS Only)

Define the CICS/DB2 connection and control characteristics using the CICS Resource Definition Online (RDO). You can still assemble macro resource control tables (RCTs) for migration to the DFHCSD file only. The DSNCRCT macro is shipped with CICS to allow migration of RCT tables to the CSD.

Note: For more information about DSNCRCT, see the *CICS Resource Definition Guide*.

The RCT or DB2 attachment facility contains the information required for CICS to establish its connection to DB2. The FILE transaction's DB2 facility lets you view information and statistics on each entry in the active RCT. The display format is presented as if the RCT existed as a DB2 table. This display is similar to the information provided by the IBM-supplied DSNC DISP STAT command, providing complete and easily accessed information.

To display the RCT, type the keyword DSNCRCT in the DB2 command area.

Versions and releases of CICS from the CICS transaction server for OS/390, Version 1 Release 3 onwards do not support running the CICS DB2 attachment facility using the RCT. Therefore, CA InterTest for CICS no longer supports the CORE=RCT command and produces the following message:

CAIN0479

KEYWORD NOT SUPPORTED IN THIS RELEASE OF CICS.

Chapter 13: Analyzing Dumps Symbolically: CA SymDump for CICS Option

What Is CA SymDump for CICS Option?

When you test a program with CA InterTest for CICS, your program does not usually produce a dump because CA InterTest for CICS monitors the program and halts it at an automatic breakpoint before the program abends. Then you can take advantage of CA InterTest for CICS's powerful diagnostic tools to find the problem, correct the error and continue testing.

Ideally, you might use CA InterTest for CICS to monitor every program and catch every error. However, in real world situations, that is not possible, due to the overhead involved or because you may not want to monitor security programs or programs using non-standard code.

Consequently, although CA InterTest for CICS greatly reduces the number of dumps, dumps will still occur, especially in production. Even a thoroughly debugged program can abend because of bad data, scheduling errors, invalid parameters, and many other factors beyond programmer control.

CA SymDump for CICS Option, which stands for symbolic dump facility, is a powerful online tool designed to handle dumps. It lets you do the following tasks:

- Analyze dumps symbolically, this is especially valuable for shooting bugs in your production system where you might not want CA InterTest for CICS breakpoint displays
- Manage your dump data set so you can selectively retain, view, and print the dumps you need, and discard the ones you do not need

Symbolic Dump Analysis

Problems that cause dumps are a fact in both production and test systems. CA SymDump Option lets you analyze and solve these problems **online**, without having to access and read traditional hexadecimal format dumps. Your source listing displays at the instruction that triggered the abend, just as if CA InterTest for CICS had stopped the program at an automatic breakpoint.

With CA SymDump Option you can do the following tasks:

- Use the powerful functions of CA InterTest for CICS to find and correct errors by:
  - Displaying program variables symbolically
  - Viewing the program listing and compiler output
  - Setting breakpoints for subsequent testing
  - Using CORE and FILE to inspect and modify main and auxiliary storage
- Take advantage of special CA SymDump Option capabilities to:
  - Pinpoint the abending instruction and its operands
  - Display the formatted trace table
  - View the source statement responsible for the abend
  - List the programs and files used by the task
  - View the contents of the PSW and registers at the abend

You can use CA SymDump Option in any region—regardless of where the abend occurred. For example, you can debug a production dump in your test region. Consequently, it is not necessary to run CA InterTest for CICS in production to take advantage of CA SymDump Option. You can even debug a dump produced in a remote data center at another site.

You can instruct CA SymDump Option to refer to the symbolics you want, if multiple symbolic versions of the program exist. This is especially helpful if there are many versions of a particular program. Even if CA InterTest for CICS does not have symbolic information for the program, CA SymDump Option still provides a formatted display of the CICS system areas, with hexadecimal and character representations of each field so you can debug with or without symbolics.

Manage Dumps

CA SymDump Option also lets you control which CICS dumps to save, discard, and print. It has its own dump data set, which is separate from the CICS dump data set. You can even route dumps to both data sets. It is easy to manage the CA SymDump Option data set because you can do the following tasks:

- **List** all dumps. You will know immediately when each dump was created and which transaction and program produced it. You can quickly select the dumps you want to view online.
- **Delete** specific dumps. You can delete the dumps you do not want and keep the ones you need. As a result, you avoid those situations where CICS stops writing dumps because the dump data set is filled.

- **Hold** specific dumps. You can specify which dumps you want to retain for future use.
- **Print** specific dumps. It is no longer necessary to print the entire dump data set to get printouts of selected dumps. You can even print an index of all the dumps in the dump data set.

How CA SymDump Option Works with Your CICS Dump Data Set

CA SymDump Option contains only application dumps; system dumps are still written to your CICS dump data set. When the CA SymDump Option is installed, your system programmers can specify which dumps they do not want written to either the CA SymDump Option or the CICS dump data set to save DASD space. For example, they can exclude all AKCT abends.

CA InterTest for CICS automatically deletes old dumps from the CA SymDump Option data set when space is needed for new ones. However, you determine how frequently dumps are deleted. You can also specify that certain dumps be **held** so they are **not** deleted. If the CA SymDump Option data set becomes full (that is, full of dumps that cannot be deleted), new dumps are written to your CICS dump data set instead of to the CA SymDump Option data set.

Learn More About CA SymDump Option

The CA SymDump Option Primary Option Menu appears when you select Option **5 Dump Analysis** from CA InterTest for CICS Primary Option menu.

Note: For detailed instructions on how to use CA SymDump Option, use the Help facility or the *CA SymDump Option User Guide.*

Appendix A: Printing the Help File

The CA InterTest for CICS Help file contains extensive tutorial information that is available to you online by entering **HELP** or pressing **PF1** from any CA InterTest for CICS screen. It has a browse capability from any point of access.

Help File Printout

You can print the Help file once it is loaded from the distribution tape using the JCL in the following examples. The program prints one screen image per page. The input to the program is the VSAM KSDS file that was loaded from the distribution tape.

- The printout is formatted as 81-byte fixed length records; the first byte contains the ANSI print control character, such as 1 for new page or a plus sign, +, to overprint the same line.
- It is produced in uppercase and lowercase to enhance readability.
- The output is over 1000 pages.
- If your printer cannot handle lowercase characters specify PARM=UPPER in the EXEC card of the JCL.

Sample MVS JCL to Print the Help File

Use the following MVS JCL to print the Help file.

```
//PRINHELP JOB....
//STEP1 EXEC PGM=IN25PRIH,REGION=256K
//STEPLIB DD DSN=YOUR.INTRTST.LOADLIB,DISP=SHR
//PROTHLF DD DSN=INTRTSTV.PROTHLF,DISP=SHR
//OUTPUT DD SYSOUT=A
//
```

Appendix B: Breakpoint Displays for NonSymbolic Programs

A non-symbolic breakpoint screen appears for all non-symbolic programs. In addition, it appears if your COBOL or PL/I program was compiled without the CA InterTest for CICS post-processor. Two examples are shown in this appendix. For more information, consult your systems programmer.

Unconditional Breakpoint

The following panel shows a sample non-symbolic breakpoint screen for an unconditional breakpoint:

```
❶                                        ❷                    ❸           ❹
CNTL=GO,TASK=00063,C             at +00BB0           pgm=COBDEMO ,addr=24510BD0
❺ UBP: F=U056,T=U056 Unconditional breakpoint.
❻Mon.as COBDEMO        ❼   Pgm COBDEMO  Trx DEMC Fac U056 Trm     in.....out
❽Task 00063 at +00BB0           ❾TIOA none                      +00A90  +00B16
 DSA 23708B88  TGT 23709D60  ❿CWK 2370A058                       +00B06  +00B16
 BLLS:  0 001000D0  1 00000000  2 00001000  3 00002000            00002  times
        4 00003000  5 00004000  6 00000000  7 00000000           +00B06  +00B24
   ①    8 00000000  9 00000000                                  +00B36  +00BA4
                                                              ② +00BA8  curr.
NEXT MVC   X'B0'(4,R8),8(R7)                 AT +00BB0  ③
PREV OI    X'84'(RD),X'01'                   AT +00BAC  ④
        Press ENTER to execute the default command on line 1,
        OVERTYPE it, or use any of the following PF KEYS:
 1 Help  2         3 Source  4         5 Resume     6 Menu  ⑤
 7       8         9 1 Inst 10        11 Backtrace 12 Status
 Addr and content of relevant area:
23708C0C 21        †        ⑥
```

❶

Default CNTL command and location information. For an automatic breakpoint, the default command abends the task. For all other breakpoints, the default is to continue task execution. The command can be modified or overtyped with any other CA InterTest for CICS command.

❷

Displacement of the instruction about to be executed follows the command on the top line.

❸

Name of the program that was stopped by CA InterTest for CICS follows the displacement.

❹

Address of the instruction that triggered the breakpoint.

Note: CA InterTest for CICS stopped the program before this instruction was executed.

❺

Type of breakpoint: automatic (ABP), unconditional (UBP), conditional (CBP), request (RBP), or single-stepping (SBP). In most cases, there is a one or two line message explaining exactly why the breakpoint occurred.

❻

Name of the monitoring table entry.

❼

Program name (Pgm), transaction code (Trx), and CICS facility (Fac) associated with the task; in this case, terminal (Trm) U056.

❽

Task number and displacement of the instruction about to be executed.

❾

Address of the first TIOA on the chain of such areas allocated to the task. To display the area, issue the command CORE=TIOA.

❿

Addresses of the areas containing the copy of COBOL DSA, TGT, Working-Storage, and Local-Storage for this task. If the program does not contain a DSA or Local-Storage, the fields are left blank. To display the area, issue the appropriate CORE command (for example, CORE=CWK).

①

COBOL BLL cells or general registers 0-15 for Assembler or PL/I programs.

②

Backtrace of the starting (IN) and ending (OUT) displacements or addresses of the most recently executed pieces of code. Every break in the sequential execution of the program results in a new entry. A loop counter tells how many times the piece of code above it was executed. The entry curr. stands for current location.

③

Machine code and offset (or absolute address) of the instruction that triggered the breakpoint.

④

Machine code and offset (or absolute address) of the last instruction executed.

⑤

Program function keys for immediate access to CA InterTest for CICS facilities.

⑥

Address and representation (in both character and hexadecimal form) of the area of main storage that would have been modified by the next instruction.

Automatic Breakpoint

The following panel shows a non-symbolic breakpoint screen for an automatic breakpoint:

```
❶CNTL=GO,TASK=00077,A            at +007A4          pgm=PL1DEMO ,addr=24517864
❷ 37 ABP: ASRA abend in user code------ data exception.           ❿

❸ Mon.as PL1DEMO         ❺Pgm PL1DEMO  Trx DEMP Fac U056 Trm     in.....out

❹Task 00077 at +007A4             TIOA none                    +00000  +00000
 DSA 23703F40                                                  +00022  +00038
❻ REGS: 0 FFFFF0F0  1 23704340  2 23704890  3 245170FC         +00014  +00020
         4 23704256  5 2451A3F8  6 2451B6F0  7 23704230         +0003C  +00358
         8 23704228  9 23704890 10 23AA4B27 11 00000000      ❼ +003CA  +00478
        12 2370BC20 13 23703F40 14 A4517754 15 00000000 CC= 0   +00480  +004B0
❽NEXT ZAP   X'A28'(4,RD),X'2F1'(3,RD)        AT +007A4         +004B8  +004C6
PREV STH   R0,0(,R1)                          AT +007A0         +004CE  +00584
        Press ENTER to execute the default command on line 1,   +051B8  +00584
        OVERTYPE it, or use any of the following PF KEYS:       +00586  +005D4
 1 Help  2         3 Source  4         5 Resume     6 Menu      +051B8  +005D4
 7       8         9 1 Inst 10        11 Backtrace 12 Status    +005D6  +005E4
 Addr and content of relevant area:                            +005EC  +00662
❾23704968 00000000                            ....             +051B8  +00662
                                                                +00664  +00692
                                                                +051B8  +00692
                                                                +00694  +00694
                                                                +00794  curr.
```

❶

Default CNTL command and location information. For an automatic breakpoint, the default command abends the task. For all other breakpoints, the default is to continue task execution. The command can be modified or overtyped with any other CA InterTest for CICS command.

Displacement of the instruction about to be executed follows the command on the top line.

Name of the program that was stopped by CA InterTest for CICS follows the displacement.

Address of the instruction that triggered the breakpoint.

Note: CA InterTest for CICS stopped the program before this instruction was executed.

❷

Type of breakpoint: automatic (ABP), unconditional (UBP), conditional (CBP), request (RBP), or single-stepping (SBP). In most cases, there is a one or two line message explaining exactly why the breakpoint occurred.

❸

Name of the monitoring table entry that caused the current monitoring to be in effect.

❹

Task number and displacement of the instruction about to be executed.

❺

Program name (Pgm), transaction code (Trx), and CICS facility (Fac) associated with the task; in this case, terminal (Trm) U056.

❻

COBOL BLL cells or general registers 0-15 for Assembler or PL/I programs.

❼

Condition Code. The value of the condition code is the value that was saved when the breakpoint occurred.

❽

Next and previous instructions. This line indicates the next instruction to execute and the instruction that was executed previously.

❾

Address and representation (in both character and hexadecimal form) of the area of main storage that would have been modified by the next instruction.

❿

Backtrace. Each pair of entries describes the boundaries of a piece of the program that executed sequentially.

Default CNTL Command

In the previous screen, the following dispatch command displayed in this field:

```
CNTL=GO,TASK=00077,A
```

This command is used to continue a program's execution from a breakpoint.

You can enter any appropriate CA InterTest for CICS command in this field. To modify this command, overtype it.

Location Information

In the previous screen, the following location information is displayed:

```
at +007A4 pgm= PL1DEMO,    addr=24517864
```

This tells you that a program named PL1DEMO is stopped at displacement 7A4. This is the displacement of the instruction that is about to be executed (check the program's listing to identify the instruction).

Reason for the Breakpoint

The type of breakpoint and the reason for it are described in the previous screen by the following message:

```
37 ABP: ASRA abend in user code- data exception.
```

- The letters ABP indicate automatic breakpoint, and the number 37 is the error code.
- The remainder of the line explains what the error code means. In this case, a data exception interrupt was set in motion by the monitored program. Ordinarily, this would cause a CICS abend with the code ASRA, but CA InterTest for CICS averted the abend with this breakpoint.

Examples

The following examples show other reasons for a breakpoint.

The following message indicates that an unconditional breakpoint (UBP) with the identification 02 occurred because the user requested it at the COBOL paragraph named 999-Abend.

```
02 UBP: At='999-abend' Unconditional breakpoint.
```

The following message indicates that an unconditional breakpoint with the identification 07 was requested by the user at the second verb of COBOL statement number 374. The following is another version of the previous message:

```
07 UBP: At=#374.1 Unconditional breakpoint.
```

If the user requests the unconditional breakpoint by address or displacement, the following message appears:

```
09 UBP: F=.ANY,T=V015 Unconditional breakpoint.
```

F=.ANY

Means that the user requested an unconditional breakpoint to occur on any terminal (or for a task that has no terminal). The parameter T=V015 means that the breakpoint display will be sent to terminal V015.

If the user requests single-stepping (SBP), the following message appears:

```
AT SBP: Single-stepping breakpoint.
```

If the user set a request breakpoint (RBP), a message similar to the following one appears:

```
RBP: Request bkpt; CICS macro is: ..............
```

If the user requests a programmed breakpoint (PBP), the following message appears:

```
00 PBP: Programmed breakpoint.
```

Monitor Table Entry

The name of the entry in the CA InterTest for CICS Monitoring Table that caused the current monitoring effort is always displayed. In the previous screen it was:

```
Mon.as PL1DEMO
```

This tells you that the program is being monitored under (MONitored AS) the entry declared explicitly for the program named PL1DEMO. It is important to know the Monitoring Table entry because the available online options are attached to that entry.

Note: As the tested task continues execution, part of it might be monitored under one Monitoring Table entry and part under another entry, depending on how you formulated the request.

Examples

If you were to request that this program be monitored only when it is run from a particular terminal, or if the online TON= option were declared, the following message would appear:

```
Mon.as PL1DEMO for U056
```

In this case, monitoring only takes place when the program named PL1DEMO is executed from the terminal named U056.

If the program was declared for monitoring under a transaction name (implicitly) rather than its program name, the following message would appear, indicating that the entry in the Monitoring Table is transaction (TRX), DEMP.

```
Mon.as DEMP TRX
```

If the program was declared for monitoring under a terminal name (implicitly), the following message would appear, indicating that the entry in the Monitoring Table is terminal (TRM), U056.

```
Mon.as U056 TRM
```

Program, Transaction, and CICS Facility

The names of the program, transaction, and CICS facility appear in the center of the third line from the top on the breakpoint screen. In the previous example, the following information is displayed:

```
Pgm PL1DEMO Trx DEMP Fac U056 Trm
```

- This tells you that the name of the program being monitored is PL1DEMO, that the transaction (Trx) is executing a program named DEMP, and that the CICS facility that generated this task is the terminal (Trm) named U056.
- For tasks created automatically by the trigger-level facility of a transient data, the FAC field names the Destination Control Table (DCT) entry.

Example

FAC GCOS DCT names the GCOS destination in the DCT.

- For tasks that do not own a terminal or a DCT entry, the FAC field displays the content of the TCAFCAAA field of the Task Control Area (TCA).

Task Identification Number and Address

The fourth line from the top of the breakpoint screen gives the task identification number and the address of the Terminal Storage Area. In the previous example, this information is expressed as follows:

```
Task 00077 at +007A4      TIOA none
```

- The parameter 00077 indicates the number of the task that was put into the breakpoint state by CA InterTest for CICS.
- The parameter at +007A4 repeats the address of the instruction that is about to be executed.
- The parameter TIOA None gives the address of the first terminal Storage Area (the Terminal Input/Output Area) on the chain of areas allocated for the task at breakpoint. You can display the first TIOA by issuing the command CORE=TIOA.

Program Storage Areas

The fifth line from the top of the breakpoint screen gives the address of various program storage areas, depending on the language of the program. For COBOL programs the line displays the address of the DSA (Dynamic Storage Area) if the program was compiled using COBOL for MVS, COBOL for OS/390, or COBOL for z/OS, the address of the TGT, the address of Working-Storage, and the address of Local-Storage, if applicable.

- The DSA address shown is the address of the Dynamic Storage Area for the current iteration of the program. To display the area, enter the command: **CORE=DSA**.
- The TGT address shown is the address of the storage area that contains the copy of the COBOL Task Global Table for this task. To display the area, enter the command: **CORE=TGT**.
- The CWK address shown is the address of the storage area that contains the copy of the COBOL WORKING-STORAGE for this task. To display the area, enter the command: **CORE=CWK**.

Important! Macro-level COBOL programs operate on one WORKING-STORAGE area that is common to all tasks.

- The LCL address shown is the address of LOCAL-STORAGE for the current iteration of the COBOL program. To display the area, enter the command **CORE=LCL**.
- For PL/I programs, the line displays the address of the program's DSA, while for Assembler programs, it displays the address of the program's work area.

BLL Cells (COBOL) or General Registers (NonCOBOL)

For COBOL programs, this area of the display shows the contents of up to 16 BLL cells used by the task at the time of the breakpoint. Each displayed fullword is preceded by a number, from 0 to 15, relative to the number of the **first** BLL cell actually used in the program.

The monitored program might be using a different number of BLL cells, but all can be displayed by issuing the proper CORE command. This can best be done using the symbolic names of the BLL cells.

Example

In the first example, the following BLL display is shown:

BLLs:	0 001000D0	1 00000000	2 00001000	3 00002000
	4 00003000	5 00004000	6 00000000	7 00000000
	8 00000000	9 00000000		

The BLL cell with the relative number 1, which in command level programs contains the address of the DFHCOMMAREA, shows the **null** value.

General Registers

For Assembler or PL/I programs, this area displays the contents of registers 0 to 15. The following example shows the 16 consecutive fullwords that CA InterTest for CICS used to save registers 0 to 15 at the time of the breakpoint.

Example

Register 0 contains the value of -3856, and registers 11 and 15 contain the null value.

REGS:	0 FFFFF0F0	1 23704340	2 23704890	3 245170FC
	4 23704256	5 2451A3F8	6 2451B6F0	7 23704230
	8 23704228	9 23704890	10 23AA4B27	11 00000000
	12 2370BC20	13 23703F40	14 A4517754	15 00000000
				CC= 0

Condition Code

The condition code appears to the right of the last line of the REGS: display, and is identified by the letters CC. The value of the condition code is the value that was saved at the time of breakpoint. For an interpretation of the condition code, consult the *IBM System/370 Reference Card.* In the previous example, the condition code is 0.

Next and Previous Instructions

The line directly below the last line of the BLLS or REGS area indicates the next instruction to be executed and the instruction that was executed previously. In the previous illustration, the following lines appears:

```
NEXT ZAP   X'A28'(4,RD),X'2F1'(3,RD)        AT +007A4
PREV STH   R0,0(,R1)                        AT +007A0
```

- In the field marked Next, the Assembler instruction that was scheduled to be executed next when the breakpoint occurred is displayed. In this example, the instruction is a ZAP. The first operand is located at displacement A28 from the value in register D for a length of 4, while the second operand is located at displacement 2F1 from the value in register D for a length of 3.

  Following the instruction, on the same line, at +007A4 gives the displacement of the instruction in the program (7A4). If the instruction resides outside the program, the address of the instruction is given instead, expressed as eight-hexadecimal digits.

- In the field marked Prev, the Assembler instruction that was executed just before the breakpoint occurred is displayed. In this example, the instruction is a STH. The first operand value is in register 0, while the second operand location is at displacement 0 from the value in register 1

  To the right of the instruction, on the same line, at +007A0 is the displacement of the instruction in the program (7A0). If the instruction resides outside the program, the address of the instruction is given instead, expressed as eight hexadecimal digits.

Data Display Area

The data display area is under the next and previous instruction line. In the previous example, it appears as:

23704968 00000000

This example shows only four bytes. However, when a larger data field is involved, the portion of the display can occupy the entire lower left corner of the breakpoint screen.

This area varies according to the type of breakpoint that occurred:

- **Automatic Breakpoint.** When possible, in the case of an automatic breakpoint, CA InterTest for CICS shows the area that would have been affected if the breakpoint had not occurred. This example shows the receiving field of the next instruction to be executed.
- **Unconditional and Single-Stepping Breakpoints.** For these types of breakpoints, CA InterTest for CICS shows the area that was affected by the previously executed instruction, where applicable.
- **Request Breakpoint.** CA InterTest for CICS does not show the data display area for request breakpoints.
- **Programmed Breakpoint.** For a programmed breakpoint, CA InterTest for CICS shows the area designated by the programmer in the CALL PBP statement that caused the breakpoint to occur.

The format of the data display area is similar to the CORE transaction screen. Each line presents up to 16 bytes in both hexadecimal and character format. On each line, the address of the first byte in that line is shown (for example, 23704968), followed by the same contents in character format between asterisks, where non-displayable values are shown as periods.

Backtrace Display

The backtrace display has two vertical columns under the heading **in.....out** on the right side of the breakpoint screen. Each pair of entries describes the boundaries of a piece of the program that executed sequentially. That is, they were executed in the same order as the machine instructions that reside in the main memory.

The backtrace display is dynamic. The most recently executed piece of code appears as the bottom line of the backtrace display, and the oldest piece of code executed is the top line. As the execution of the program progresses, new lines are added at the bottom and dropped from the top.

The following screen repeats the backtrace display shown in the previous example.

```
    in.....    out

     +00000   +00000
     +00022   +00038
      +00014  +00020
      +0003C  +00358
      +003CA  +00478
      +00480  +004B0
      +004B8  +004C6
      +004CE  +00584
      +051B8  +00584
      +00586  +005D4
      +051B8  +005D4
      +005D6  +005E4
      +005EC  +00662
      +051B8  +00662
      +00586  +005D4
      +051B8  +005D4
      +005D6  +005E4
      +005EC +00662
      +051B8  +00662
      +00664 +00692
      +00586  +005D4
      +051B8  +00692
      +00694  +00694
      +00794  curr.
```

In the previous example:

- The letters curr. stand for current location. The current location is also shown in the first line of the breakpoint screen (+00794 curr is the most recently executed piece of code).
- The # character precedes each statement number.
- Displacements or addresses are given for lines whose statement numbers could not be found. The displacements are preceded by a plus sign, +, and the addresses are expressed as six-hexadecimal digits.

The backtrace display reveals a number of facts about the way the monitored program behaved:

- The bottom line, for example, shows that from +00794 to curr (+007A4) the program executed sequentially.
- Before this happened—as the fourth line from the bottom shows—the program executed its instructions sequentially (from +00664 to +00692), but then +00692 passed control to the +051B8, which returned control back to +00692. Most likely, this means that at +00692 the program issued a CICS command.

In this manner, you can trace, with great precision, the execution of the program from any point in its logic all the way up to the breakpoint.

Notes:

- If one of the pieces of code executed a loop, the number of times the loop was executed appears as a decimal number under the appropriate line in the backtrace display.
- If the **,GO= element** of the CA InterTest for CICS dispatch command was requested, the change in the logic flow is shown in the backtrace display just as if the program executed a branch or a GO TO command.
- At the beginning of the execution of a COBOL program, a number of lines of the backtrace display only indicate the execution of the COBOL initialization routines. The first statement number shown is the one from which the first branch instruction was executed. This limitation should not pose a problem, since the execution of the Procedure Division began from the first coded statement.

Appendix C: Examining Dumps

Two kinds of *abends,* or abnormal terminations, are of interest to CA InterTest for CICS users:

- Abends in CICS during execution of a transaction

  When this happens, you must examine the transaction dump associated with the abend to find any CA InterTest for CICS Work Areas (also known as CA InterTest for CICS Diagnostic Areas). The presence of such an area is a sure indication of CA InterTest for CICS monitoring. Only in this area can you find certain data about the program, such as the values of registers, that you will need to know.

- Abends in CA InterTest for CICS transactions (CNTL, CORE, FILE, ITST, LIST, SYMD)

  To understand why these transactions abend, you need to know how they work in CICS:

  - CNTL and CORE transactions, when necessary, follow the chain of active CICS tasks and the chain of suspended CICS tasks to locate the task whose data is to be displayed, and any other tasks to which the CNTL command could apply. If a chain is damaged, these transactions will abend. Yet, the CICS system might be able to function for some time with these chains damaged, as long as the CICS software does not have to scan all the chains.

  - CORE and FILE transactions are conversational, and can be defined in the CICS Program Control Table (PCT) so that they will be abended by CICS if you take too long to respond with the next command.

  - FILE transaction is simply a generalized application program, and it can abend in situations where the parameters you submit could not be checked sufficiently before the CICS command was issued. In most instances, CA InterTest for CICS does not attempt to check for the same things that the CICS software checks for, because such a duplication of effort would significantly increase the size of the CA InterTest for CICS software; even if those conditions were discovered by the CA InterTest for CICS programs, CA InterTest for CICS would have to issue the abend.

CICS Abend Codes

CA InterTest for CICS issues the following error codes when error conditions are discovered:

CORE—Is the same as the transaction code of the CORE transaction at your site. It means that the CORE transaction intercepted a CICS abend that makes it impossible to continue the transaction.

INHA—Indicates that a program (usually a user program) has corrupted a CA InterTest for CICS storage area so that CA InterTest for CICS cannot continue processing. Check the CICS statistics to see if any storage violations occurred. If you find any, see which program caused the violation. This would be a program not being monitored by CA InterTest for CICS.

INTE—Indicates that an automatic breakpoint cannot be executed because:

- The CA InterTest for CICS automatic breakpoint facility is disabled.
- There is no terminal on which to show the breakpoint display.
- The terminal designated to receive the breakpoint display is not a 3270-type CRT (or compatible model).

INTP—Indicates that a PLI LE program was linked without CEESTART as its first CSECT and CA InterTest abends the task with an INTP abend code.

KERN—Indicates that the task at a breakpoint could not recover from an abend. If you know the reason (for example, you abended the task), ignore this code; otherwise, contact CA InterTest for CICS technical support.

NATI—indicates that a breakpoint display is being directed to a terminal that does not have the Automatic Task Initiation (ATI) capability. If the status of the terminal is either transceive or receive (preferably, transceive), the terminal has the ATI capability. You can adjust the status by the CEMT service transaction.

Diagnostic Area

CA InterTest for CICS obtains the CA InterTest for CICS Diagnostic Area when it begins monitoring a program. CA InterTest for CICS keeps the area until the program finishes its execution in that task.

There is one such area per *logical level* of a monitored program, per task. This means that in most cases there is only one area. However, if you have a situation in which program A links to program B, which then links to program A again, the second use of program A necessitates a second CA InterTest for CICS Diagnostic Area for the same program.

The CA InterTest for CICS Diagnostic Area is in a User class of storage, on the task's storage chain; therefore, it is printed in the CICS transaction dump. In the dump, you can identify the CA InterTest for CICS Diagnostic Area by the characters **InterTest for CICS** at displacement 8 in the **User Class** storage area (hexadecimal 8C at displacement 0).

You should first look for a CA InterTest for CICS Diagnostic Area in any transaction dump and make sure that the area was obtained for the program in question. Whatever the reason for the dump, the only place you can find the registers and the most recent instruction is in that area. If more than one area exists for the program, the one with the highest value in the WKACLOCK field (displacement 0260, hexadecimal) is the most recent one.

The following table provides an Assembler-like description of the fields in the CA InterTest for CICS Diagnostic Area that are used in debugging. The displacement of each field is given in hexadecimal on the left.

Release 4.x / 5.x	Field	Type	Description
0000	WKAREA	DSECT	
0000	WKSAA	DS 8X	Storage Accounting Area (contains X'8C' in the first byte)
0008	WK1D1	DS CL9	If the 'InterTest' identifier is corrupted, the area is not valid
0011	WKTASK#	DS PL3	Task identification number
0014	WKTRANS	DS CL4	Transaction identification
0018	WKPGMID	DS CL8	Program identification
0020	WKTERM	DS CL4	Terminal identification
0024	WKDATE	DS PL4	Date
0028	WKTIME	DS PL4	Time
002C	WKMESG	DS 0H	Abbreviated error message; occurs when an INTE abend was issued instead of an automatic breakpoint
002C	WKERCODE	DS C' XX'	Error code, hexadecimal If you requested an abend when the task was at breakpoint, this code will be 00. Also, see displacement 0248.
0030	WKERTEXT	DS C'INTE	C'INTE ABEND AT '
003E	WKERDSPL	DS C' XXXX.'	Displacement, at the abend request, of the instruction to be executed
0044	WKINSR	DS CL6	Instruction about to be executed
004C	WKINSRAD	DS XL4	Address of the instruction in WKINSR

Release 4.x / 5.x	Field	Type	Description
0050	WKTODO	DS CL6	Instruction to be executed, if the instruction in WKINSR is an EX (execute) instruction
0058	WKTODOIA	DS F	Address of instruction in WKTODO
005C	WKTODOAD	DS F	Address of storage area to be affected
0060	HISWKSAV	DS CL64	Registers 0 – 15; values of the registers shown before execution of the instruction in WKINSR. This is the best place to look for values in the registers, even if the abend was not issued by CA InterTest for CICS.
00A0	WKWITHIN	DS CL256	Area surrounding the address that is about to be affected by the instruction in WKINSR; valid only if the error code is not zero and an IN25 abend occurs.
01A0	MYSAVE	DS CL64	Internal CA InterTest for CICS Save Register Area
01E0	CURHISP	DS F	Address of the current instruction
01E4	NEXHISP	DS F	Address of the next instruction
01E8	CONDCODE	DS F	Condition code of application program. The condition code is saved in the left-most byte of this field, in the same format as in the PSW.
01EC	HISBEG	DS F	Beginning of monitored program
01F0/Not available	WKACPPTE	DS F	PPT entry of PGM in control
0204/254	WKPROMHD	DS F	Address of Monitoring Table header
0208/258	WKADSOCB	DS F	Pointer to active ADS OCB
0214/264	WKPROMPT	DS F	Address of Monitoring Table entry being used for monitoring
0218/268	WKARNEXT	DS F	Next IN25 WKAR
021C/26C	WKARPREV	DS F	Previous IN25 WKAR
022C/274	WKPROMAI	DS F	Pointer to Main PROM Table Entry
0244/290	XTENTABA	DS F	Address of backtrace table that records extents, or pieces of code executed sequentially with no branch instructions taken, of the monitored program
0248/294	ADSMSLT1	DS C'ABP XX'	Secondary error or breakpoint message; *XX* is the CA InterTest for CICS error code. If task abended on your request from an automatic breakpoint, the ABP error code, displacement, and address are found here. Instead of 'ABP XX', the message could say 'UBP XX' where *XX* is the ID number of the unconditional breakpoint.

Release 4.x / 5.x	Field	Type	Description
024F/29B	ADSMSDSP	DS C'XXXX'	Displacement
0253/29F	DS	C' '	
0254/2A0	ADSMSADR	DS C'XXXXXX'	Address
025A/2A6	DS	C' '	Contains the reason for a breakpoint if the task abended with a dump after a breakpoint

ABP and INTE Abend Error Codes

The CA InterTest for CICS Diagnostic Area is printed as one of the user class storage areas in the INTE abend dump. It contains the error code as the first two characters of message XX INTE ABEND..., which can be found at the beginning of the area. The codes and their messages are shown in the following table. These two-character hexadecimal codes appear in the Inquiry Repor*t*, which can be obtained by pressing PF3 from the Breakpoint Display, or by entering CNTL=INQ.

Code	Description
01	Assembler only; an EX machine instruction issued upon an EX instruction.
02	An invalid machine instruction code. Note no override by OVR option.
03	This machine instruction not permitted under CICS; for example, an SVC.
04	A privileged machine instruction, illegal under CICS.
05	An operating system GETMAIN request, as opposed to a CICS GETMAIN.
06	The storage area about to be affected is, entirely or in part, outside of the storage that is legally available to this CICS application.
07	The storage area about to be affected resides outside of the CICS region.
08	The named program module cannot be found in the program table (PPT), its PPT entry is disabled, or it is unavailable to this CICS application.
09	The program module about to be deleted already has a zero current use count.
0A	The Transient Data or Temporary Storage area is incorrectly specified.
0B	This Terminal Control request incorrectly specifies the TIOA address.
0C	The storage area specified for this CICS request does not belong to this task or is not a CICS storage area.
0D	CA InterTest for CICS logic error; call CA Technical Support for assistance.

Code	Description
0E	This FREEMAIN request is illegal or specifies a wrong storage area address.
0F	With this task about to terminate, one or more RELOAD=YES program modules still do not have their main storage areas FREEMAINed.
10	Possible incorrect length of a temporary storage or transient data record.
11	A wild branch or direct machine instruction branch outside of the program module was detected. If not a mistake, apply the FOL option and continue.
12	Macro level coding only; this CICS macro specifies incorrect storage area.
13	A GETMAIN request with FLENGTH for more than 65504 bytes can be issued only when the program is in AMODE=31.
14	An attempt to modify this load module, which is considered reentrant, either explicitly (RNT=ON option) or as PL/I or COBOL II.
16	A GETMAIN for 0 bytes of storage has been detected.
17	Macro level coding only; incorrect length field in a variable-length record.
18	Macro level coding only; the DFHFC macro does not specify an FWA.
19	Macro level coding only; the DFHFC macro does not specify the storage area of CLASS=FILE.
21	CA InterTest for CICS ordered to continue a task from a breakpoint at an incorrect location.
23	The total number of CICS requests this program issued while monitored in this task exceeded the value specified in the MXR option.
24	The total size of main storage this task is using just exceeded the value specified in the MXS option.
25	CICS abend intercepted, but not immediately after the last CICS request (either macro or command) or another outside CALL recognized by CA InterTest for CICS.
26	CICS abend intercepted while a CICS macro was in progress.
27	CICS abend intercepted while an EXEC CICS command in progress.
28	CICS abend intercepted while a CALL outside the program, such as a Database request, recognized by CA InterTest for CICS, was in progress.
29	EXEC CICS RETURN with TRANSID issued from other than the task's top logical level.
2C.	An abend occurred while processing an IN25UEXI routine.
2D.	A program check occurred in IN25PGM2.
34	A protection exception program check caused an ASRA abend intercept.

Code	Description
35	An addressing exception program check caused an ASRA abend intercept.
36	A specification exception program check caused an ASRA abend intercept.
37	A data exception (invalid packed numeric data format) caused an ASRA abend intercept.
38	A fixed point overflow program check caused an ASRA abend intercept.
39	A fixed point divide program check caused an ASRA abend intercept.
3A	A decimal overflow program check caused an ASRA abend intercept.
3B	A decimal divide program check caused an ASRA abend intercept.
3C	An exponent overflow program check caused an ASRA abend intercept.
3D	An exponent underflow program check caused an ASRA abend intercept.
3E	A significance program check caused an ASRA abend intercept.
3F	A floating point divide program check caused an ASRA abend intercept.
CC	The storage area about to be affected by this EXEC CICS command resides, all or in part, outside the CICS main storage legally available to this application program. The area could have been specified as INTO ( ), SET ( ) or one of the LENGTHs.

Execution Extents Table (Backtrace)

CA InterTest for CICS records the following information in the Execution Extents Table as it monitors a program:

- Pieces of the monitored program's code that were executed sequentially, without any branch instructions taken
- Number of times the loop was executed, if the code contains a loop
- Each CICS command and macro request
- Each CALL to a supported database management system

This information is on the breakpoint display as the backtrace. See the appendix, "Breakpoint Displays for Non-Symbolic Programs" for a description of the breakpoint display and an explanation of the backtrace. Whenever possible, CA InterTest for CICS converts COBOL program displacements into COBOL statement and verb numbers **before** the display is shown.

There is one Extents Table per CA InterTest for CICS Diagnostic Area, pointed to by the address at displacement X'244' for Releases 4.x, and displacement X'0290' for Releases 5.x. The field name is XTENTABA. The table is a user-class storage area. It can also be identified by the characters BKTR near the beginning of the storage area.

The table consists of six-byte entries. You can find the oldest entry at the four-byte address that immediately follows the BKTR identifier. Each entry contains three halfwords, described as follows:

- The first halfword contains the displacement of the first instruction in the extent.
- The second halfword contains the displacement of the last instruction in the extent of the token branch.
- The third halfword contains the binary number of times the extent was executed in a loop.

If the lower-order bit (X'0001') is on in either of the first two halfwords, the table entry does **not** describe an Execution Extent. If execution is being traced outside the monitored program, or if the displacement s= is greater than or equal to 64 KB, the displacement is invalid.

Entries in the CICS Trace Table

Special Entries

CA InterTest for CICS writes special entries into the CICS Trace Table to assist you in your analysis efforts. These entries appear in the Trace Table when it is found in a dump, and when it is displayed with a CORE=STRA command (at a breakpoint) or the CORE=TRT command.

In the CICS formatted dump, these special entries are identified as follows:

```
USER 195.
```

Unlike the regular CICS-written entries, the CA InterTest for CICS entries provide displacements, which are the same displacements you can find in the listing of the program. CA InterTest for CICS writes these entries into the Trace Table, in addition to those issued by the monitored program and the CICS software, when the monitored program is about to:

- Issue a CICS request (command).
- Issue a CICS request (macro).

- Be stopped at a CA InterTest for CICS breakpoint or resume execution after a CA InterTest for CICS breakpoint.
- Execute an instruction for which CA InterTest for CICS disregards a presumed error condition because of an OVR (Override) option in effect.
- Issue a DL/I request.
- Issue a CICS request (either command or macro) and the active online NUP= option will affect the request.
- Issue a call to software defined in the CA InterTest for CICS IN25UEXI table.

The entries issued for these cases consist of two fields, which are described next.

- Field A contains the displacement of the request (or the breakpoint) from the beginning of the program, expressed as four hexadecimal digits. If the request was made outside the program, this field contains the address instead.
- Field B contains the following bytes:
  - At a CICS command, bytes 0 to 3 contain four hexadecimal digits that specify the command code (such as, 0204 for a handle condition).
  - At a CICS macro, bytes 0 to 1 contain two characters that describe the service type (SC = storage control, FC = file control, and so on) and bytes 2 to 3 contain the request field in the TCA.
  - At a breakpoint, bytes 0 to 2 contain the code indicating the type of breakpoint (ABP = automatic breakpoint, UBP = unconditional breakpoint, and so on) and byte 3 contains either a B (for Before the breakpoint) or an A (for After the breakpoint).
  - At each override condition, bytes 0 to 2 contain the characters OVR and byte 3 contains the hexadecimal error rode. (The OVR option cancels the Automatic Breakpoint display.)
  - At a DL/I request, bytes 0 to 1 contain the characters DL.
  - At an NUP-affected request, bytes 0 to 2 contain the characters NUP (the online NUP= option prevents updates to files).
  - At a call to CA InterTest for CICS-recognized software, bytes 0 to 1 contain the characters US and bytes 2 to 3 contain the numeric identifier from the IN25UEXI table.

CA InterTest for CICS also writes four entries into the Extents Trace Table when the monitored program is about to pass control directly to another program (by a branch instruction and **not** by a CICS command or macro) **and** the online FOL= option is in effect. These entries are as follows:

- In field A, the first entry contains the displacement that passes control. In field B, it contains the characters FROM.
- In fields A and B, the second entry contains the name of the program that passes control.
- In field A, the third entry contains the displacement that obtains control. In field B, it contains the characters INTO.
- In fields A and B, the fourth entry contains the name of the program that obtains control.

When monitoring is dropped because of a BYP option or a request by an IN25UEXI routine, field B contains DROP.

InternalProcessing Entries

CICS trace table entries for CA InterTest for CICS internal processing do not appear unless you request them. Normally, you do not want them when you are analyzing your application's processing; the CA InterTest for CICS program entries might conceal your application program entries.

To place the CA InterTest for CICS program entries on the CICS Trace Table, issue the command:

```
CNTL=ITTRACE,ON
```

To remove the entries, issue the command:

```
CNTL=ITTRACE,OFF
```

To place the CA Intertest for CICS common user interface entries on the CICS trace table and log the TCP/IP message packets to the MSGUSR DD issue the following command:

```
ICDB TRACE_ON
```

To remove the entries, issue the command:

```
ICDB TRACE_OFF
```

Note the following:

- Turn on the ITTRACE before creating a dump for Computer Associates Technical Support Staff.
- These commands have no effect on the CICS Trace Table entries normally made by the monitored program or the special entries made by CA InterTest for CICS, which is discussed in the previous section.

Appendix D: Using the FOL= Option

Note: Although the FOL= option can still be used, it was replaced by Source Listing and CNTL facilities to dynamically called programs. It is often difficult to debug software that does not follow CICS standards. To use CA InterTest for CICS on such software, you must request the online FOL= option, which enables CA InterTest for CICS to monitor a program even when that program passes control to an unknown address (for example, an address that is not part of CICS or another interface supported by CA InterTest for CICS).

The chapter "CNTL: Commands and Menus" explains how to set and remove the FOL= option. This appendix provides additional background information on how the FOL= option is actually used by the CA InterTest for CICS system.

For more information:

Source Listing Facility (see page 47)
CNTL Commands and Menus (see page 295)

Find the Program Table Entry

When the FOL= option is in force and the control of the CPU is about to leave the code of the monitored program, CA InterTest for CICS attempts to find out the following:

- What program is about to receive control
- What options, such as breakpoints, apply to that program

CA InterTest for CICS does this by trying to find the CICS Program Table (PPT) entry belonging to the program that is about to receive control. If CA InterTest for CICS can locate this entry, it is possible for CA InterTest for CICS to use displacements that apply to the beginning of the new program rather than displacements that apply to the old program.

If the PPT entry is found, CA InterTest for CICS does the following:

- CA InterTest for CICS makes four special entries in the CICS Trace Table to indicate which program passed control and from where it passed control, and which program received control and from where.
- CA InterTest for CICS begins to show data pertaining to the new program in the breakpoint display. For example, any displacements in the backtrace portion of the breakpoint screen are calculated relative to the program that just received control. You also see a short message in the backtrace display to indicate which program passed control directly to the current one.

The PPT entry is only found if the target program is located, or if it resides in main storage. If the program is not permanently resident, it is considered only if its current use count is at least one. In addition, CA InterTest for CICS will not be able to find the PPT entry if control is being passed to a piece of code that does not reside in a storage area of the CICS Program Storage Class.

Find the Monitoring Table Entry

If the new PPT entry is found, CA InterTest for CICS also tries to find the Monitoring Table entry that belongs to the program that got control directly. This lets the new PPT entry assume all options, such as breakpoints, that were declared for the program that is about to receive control.

If found, the new Monitoring Table entry is assumed to be together with all the options associated with this entry. However, CA InterTest for CICS carries the FOL= option over such a Monitoring Table switch so that the FOL= option remains active for the remainder of the monitoring effort, even though the Monitoring Table entry that is now in control might **not** have a FOL= option declared. This means that any options, such as breakpoints, that you want to see applied to the program that is receiving control can be declared for the Monitoring Table entry associated with the program.

Moreover, you can declare these options in the most convenient way; for example, breakpoints can be declared by displacements, and the UBP= and BYP= options can be declared for displacements and for addresses. The addresses ensure that if CA InterTest for CICS cannot find the PPT and Monitoring Table entries, it continues monitoring under the Monitoring Table entry of the program that passed control.

Important! CA InterTest for CICS disregards the Exclusion Table during the search for the new Monitoring Table entry because monitoring cannot be interrupted at the place of the wild branch. Before CA InterTest for CICS can interrupt monitoring, it has to know where control is returned so that later it can resume monitoring. However, the location to which control is returned depends on the design of the monitored program, and CA InterTest for CICS cannot predict that. This is why monitoring cannot be interrupted on a wild branch.

Appendix E: Monitoring Restrictions

This appendix discusses the monitoring restrictions that apply to CA InterTest for CICS.

This section contains the following topics:

Program Control Restrictions (see page 573)

Program Control Restrictions

For CA InterTest for CICS to be able to monitor a program, it must receive control from the CICS EXECUTE INTERFACE PROGRAM (DFHEIP). During startup CA InterTest for CICS enables the global user exit points XPCFTCH and XPCABND provided by IBM. These points allow a program to assume control of another program during an EXEC CICS LINK, an EXEC CICS XCTL, or abend processing. To verify the status of these exits, use the IGLU transaction.

If a program is eligible for monitoring by CA InterTest for CICS, monitoring can begin only if one of the following conditions is satisfied:

- The program to be monitored is the first program to participate in a transaction. In other words, the program was named in the CICS Transaction Resource definition.
- The program to be monitored received control from another program through an EXEC CICS LINK or EXEC CICS XCTL.
- The program to be monitored received control from another program through a direct branch instruction. In addition, the program that passed control must be monitored by CA InterTest for CICS, and must have the online FOL=option active. For more information, see the chapter "CNTL Commands and Menus."
- The program to be monitored received control from another COBOL program using a dynamic call.
- The program to be monitored received control as the abend exit routine. In other words, the program (or part of it), was named in an EXEC CICS HANDLE ABEND command, executed prior to the abend.

If a *non-monitored* program passes control to another program directly through a branch instruction rather than an EXEC CICS LINK or EXEC CICS XCTL command, monitoring is *not* set in motion. This is true even though the program that is receiving control is declared in the CA InterTest for CICS monitoring table. In order to cause monitoring to start in this situation, segmented monitoring may be used or the program issuing the direct branch must be monitored with the FOL=option declared. For more information, see the FOL= keyword of the IN25OPTS macro.

Execution Restrictions

As CA InterTest for CICS executes the instructions of the monitored program, it must be able to recognize the following requests:

- Any CICS service requests. This is to prevent CA InterTest for CICS from monitoring the CICS software.
- Any other means the monitored program could use to pass control to other parts of the application.

CA InterTest for CICS recognizes CICS service requests if the commands are coded into the monitored program. However, it might not recognize calls to the system if the statements that prepare the service request are coded and arranged differently than as generated by the CICS command translator.

Special Circumstances

You should be aware of the following special circumstances:

- CA InterTest for CICS will not monitor a program whose name begins with the characters DFH or IN25 unless that program is specifically declared for monitoring by its program name. Such a program cannot be monitored by declaring a transaction or terminal for monitoring, or even by declaring the entire application system: CA InterTest for CICS automatically assumes this program belongs to the CICS system software.
- You must prevent CA InterTest for CICS from monitoring certain programs that run under CICS, such as TRUE's, GLUE's, and any user replaceable system components (such as DFHPEP). Use the CNTL=EXCL command to exempt these programs from monitoring. You can identify the program, transactions, and terminals to be exempted by giving their full identifications or by using generic identifications. However, it is best to follow the procedure CA recommends and always explicitly declare programs for monitoring by using a CNTL=ON, PROG=... command or by specifying specific programs in the menu system.
- CA InterTest for CICS automatically suppresses breakpoints between an EXEC CICS POST and EXEC CICS WAIT EVENT, WAIT EXTERNAL, WAIT CONVID, or WAITCICS. When the EXEC CICS WAIT is encountered, breakpoint processing resumes. Starting the CA InterTest for CICS internal VTAT transaction, when a breakpoint is encountered, would freemain the ECB set by the EXEC CICS POST causing the EXEC CICS WAIT to hang waiting on the ECB. Therefore, CA InterTest for CICS suppresses any type of breakpoint set between POST and WAIT allowing the application program to continue normally.

Index

=

=x.y.z command • 57
  on Source Listing display • 57

A

abending tasks • 307, 372
abends, CICS, codes, CA InterTest for CICS • 560
ABP • 252, 270, 309, 315, 363, 371
  keyword • 309, 371
  option • 252, 270, 315, 363
action characters, Source Listing display • 63
active tasks • 263, 271
  disconnecting from terminal • 263
  purging • 263
  viewing online • 271
AICA abends, preventing updating • 319
ALL keyword • 302, 313, 371
application invocation parameter list • 537
automatic breakpoints • 136, 143, 145, 240, 252, 269, 270, 273, 302, 309, 315, 324, 341, 363, 371
  global facility • 269, 309, 371
  HELP for correcting/continuing • 143
  local option • 252, 270, 315, 363
  monitoring, automatically setting • 240, 302, 341
  non-3270 terminal tasks • 273
  overriding • 145, 324
    OVR option • 324
    permanent • 145
    temporarily • 145
  production system • 252, 269, 270, 315, 363
  source listing example • 136
automatic single-stepping • 82, 372
  resume task execution • 372
  stop value • 82
  turning on • 82
  wait amount • 82
auxiliary storage • 29, 441

B

Backtrace facility • 167, 174, 181
  accessing • 167
  automatic stepping • 174
  bookmark • 167
  navigating through execution • 174
  overview • 167
  Source Listing Breakpoint screen • 167, 181
Backtrace Summary • 167, 170, 171
  bookmark • 167
  From field • 167
  PF keys • 171
  reading • 170
  select column • 167
  Source Listing Summary • 167
  statement block • 167
  view • 167
BACKWARD command • 41
BALR instructions, permitting • 249
BDAM files • 503, 504, 505, 506, 507, 508, 509
  adding a record • 509
  browsing a file • 506
  FILE functions • 504
  FILE screen • 503
  searching for a data string • 507
  updating a record • 508
  viewing a record • 503, 505
BKPT Terminal ID, changing from Source Listing Profile • 84
bookmark • 167
BOTTOM command • 41
Breakpoint Locations menu • 110, 112
  example • 112
  stop at all data name references • 110
  stop at all paragraphs or all labels • 110
  stop at first statement • 110
  stop at specified verb • 110
Breakpoint Locations screen • 109, 131, 342, 346
Breakpoint Primary Option Menu • 37, 40, 163
  menus and submenus • 40
  overview • 163
  using • 37
  ways to access • 37
Breakpoint-Related Areas menu • 390, 397, 399, 401
  Assembler • 399
  COBOL • 390, 397
  PL/I • 401
breakpoints • 24, 27, 103, 105, 109, 146, 226, 249, 263, 272, 273, 304, 309, 361, 371, 372, 390, 391, 392, 397, 398, 399, 401, 412, 413, 549, 550, 552, 555, 556

activities • 27
CORE commands • 412, 413
CORE menus • 390, 391, 392, 397, 398, 399, 401
Assembler • 399
COBOL • 390, 391, 392, 397, 398
PL/I • 401
data, working with • 146
definition • 24
disabling • 372
instructions, bypassing • 249, 361
moving • 304, 372
non-symbolic • 549, 550, 552, 555, 556
non-terminal tasks • 272, 273
purging tasks • 309, 371
resetting after recompiling • 226
scrolling to next • 226
setting • 27
setting unconditional using Breakpoint Locations screen • 109
setting, when, where and why • 103
terminals, disconnecting at • 263
transferring with New Program Copy • 105
BTRACE command • 167
BWD command • 41, 57
description • 41
on Source Listing display • 57
BYP option • 249, 361
Bypass of automatic breakpoint • 145

C

CA InterTest for CICS • 23, 24, 30, 33, 47, 48, 68, 70, 71, 72, 148, 184, 185, 301, 308, 309, 370, 371
definition of • 23
global logging • 309, 371
initializing • 47, 68, 70, 71, 72, 148, 184, 185, 301
monitor • 24
reports • 308, 370
starting • 47
Symbolic file • 308, 370
tables • 308, 370
terminating • 301
transaction IDs • 48
transactions • 33
usage methods • 30
CA SymDump for CICS • 539, 541
access from CA InterTest for CICS • 541
analyzing dumps • 539
managing dump data sets • 539
overview • 539
CAIN8000, error resolution • 94
CAIN8001, error resolution • 94
CAIN8002, error resolution • 94
CAIN8003, error resolution • 94
CAIN8004, error resolution • 94
CAIN8005, error resolution • 94
CBP option • 316
changing the From Terminal ID • 84
changing the terminal ID • 84
Checkpoint facility • 309, 371
CHGE function • 461
CICS • 125, 252, 355, 363, 379, 401, 405
areas • 379, 401, 405
displaying (CORE menus) • 401, 405
structured storage display • 379
commands, halting program before • 125, 355
limiting requests • 252, 363
macros, halting program before • 125, 355
CICS Life of Task • 535
CKPT keyword • 309, 371
CLOT keyword • 535
CNTL • 295, 333, 334, 335, 340
capabilities • 295, 333
functions • 335
HELP • 334
menus and screens • 335
PF keys • 340
CNTL commands • 57, 68, 70, 71, 72, 148, 184, 185, 295, 297, 299, 300, 301, 302, 303, 304, 305, 307, 308, 309, 311, 313
abending tasks • 307
generic specification • 299
hierarchy rules • 300
initializing ?CA InterTest for CICS • 184
initializing CA InterTest for CICS • 68, 70, 71, 72, 148, 185, 301
modules of • 311
monitoring • 302, 304, 309
excluding • 304
removing • 302
resetting • 309
setting • 302
on Source Listing display • 57
options • 302, 303, 311, 313
removing • 302, 313
setting • 302, 303, 313
summary of • 311
syntax • 313

reports, status • 308
restarting CA InterTest for CICS CA InterTest for CICS • 301
restarting • 301
resuming task execution • 305
resuming task execution at an indirect command • 305
resuming task execution GO keyword • 304
summary of • 295
syntax • 297
terminating CA InterTest for CICS • 301
CNTL commands, system-wide options, setting • 309
CNTL Function Selection menu • 280
COBOL II • 91, 268
dynamically called programs • 268
LE/370 condition handlers • 268
nested programs • 91
using the post-compiler • 91
COBOL II and COBOL/370 • 571
COBOL/370, dynamically called programs • 268
COLLAPSE command • 41
command field, commands supported from Selection Lists • 41
compiler output, access to • 32
composite support • 98, 255, 256, 257, 259, 260, 262
duplicate subprograms • 98
example of • 259
monitoring • 255
removing • 262
setting (Assembler) • 262
setting (batch) • 256, 257
setting (online) • 260
Conditional Breakpoint screen • 115, 119, 121, 347, 350, 352
COBOL • 115, 119, 347, 350
detailed • 121, 352
conditional breakpoints • 27, 113, 115, 119, 121, 122, 123, 132, 316, 346, 347, 349, 350, 351, 352, 353, 354
Assembler and PL/I • 121, 352
COBOL • 115, 347
CORE keywords • 123, 354
defined • 27
example • 350
figurative constants • 119, 349
length of comparision • 122, 353
removing • 132, 316, 346, 351, 352
setting • 119, 132, 316, 346
Source Listing facility, setting • 113
Conditional Locations screen • 132, 351
COBOL • 132
remove breakpoints • 351
conventions, fastpath menu notation • 38
CORE • 375, 376, 377, 379, 381, 385, 387, 388, 389
dump format • 385
menus • 375
menus, accessing • 388
overview • 375
PF keys • 387, 389
menus • 389
structured display • 387, 389
Source Listing Breakpoint screen • 375
storage display formats • 376
structured display • 377, 379, 381
Assembler and CICS • 379
COBOL • 377
PL/I • 377
USE option • 381
CORE commands • 57, 381, 406, 407, 409, 410, 411, 412, 413
elements • 409
indirect addressing • 411
locations and displacements, specifying • 409
main storage • 407, 412, 413
displaying • 407
displaying at breakpoint (COBOL and PL/I) • 412, 413
on Source Listing display • 57
overview • 406
scrolling • 411
storage pointer • 409, 410
USE option • 381
CORE keywords, conditional breakpoints • 123, 354
CSA option • 249, 317, 318, 361
customization, EDF • 291
CWA option • 249, 317, 318, 361

D

data • 100, 146
breakpoint activities • 146
monitoring • 100
data name, qualified name • 91
databases, password-protected • 456
DATATYPE field, temporary storage • 511
DB2 databases • 488, 489, 490, 494, 495, 496, 497, 498

columns, selecting • 490, 494
command list • 498
confirming changes • 497
deleting • 495
FILE functions • 494
FILE screen • 488
FILE, accessing • 488
rows, selecting • 496
scrolling • 489
SQL commands, entering • 489
updating • 495
DB2 programs • 524, 526, 527, 528, 532, 534
advanced debugging • 532
attach facility • 524
calls, types of • 532
DSNC abends • 534
host variables, inspecting • 527
SQL requests, breakpoints for • 526
SQL return codes • 528
demo Sessions, available in Primers • 47
detailed breakpoint • 176, 178, 181, 182, 183, 186
additional displays • 176
backtrace • 178
CNTL commands, defaults • 183
execution, continuing • 182
PF keys • 178, 181
reading a backtrace • 181
restoring the breakpoint • 186
source code view options • 186
switching to Source Listing • 186
using • 178
DFLTUSER, viewing value online • 232
diagnostic area, CA InterTest for CICS • 560
diagnostic information • 26
display • 77
registers • 77
titles • 77
DL/I databases • 125, 355, 477, 480, 483, 484, 485, 487
FILE functions • 480
FILE screen • 477
halting program before calls to • 125, 355
segment • 483, 484, 485, 487
adding • 484
copying • 485
deleting • 487
updating • 483
SSA formats • 480
DM option • 318
DM option Statement Trace options • 357
DM option • 357
DOWN command • 41
DSNC abends • 534
DSNCRCT keyword • 498
dump analysis • 539
dumps • 539, 540, 541, 559, 560, 565, 566, 568
analyzing • 539
CICS data set • 541
examining • 559, 560, 565, 566, 568
managing • 540
symbolic display • 539
duplicate subprograms, composite support • 98
dynamically called programs • 571
monitoring • 571

E

EDF • 291
END command • 41, 57
ISPF commands • 41
on Source Listing display • 57
END keyword • 301
ending FILE session • 457
errors • 25, 324
detection and prevention • 25
overriding • 324
EXCL keyword • 304
Execution Extents Table • 565
execution path, tracing • 167
EXPAND command • 41

F

FEP option • 318
FEPI, Front-End Programming Interface • 290
FILE • 161, 162, 441, 443, 445, 455, 456, 457, 458, 460, 461, 462, 464, 465, 466, 477, 487, 503, 505, 506, 507, 508, 509, 511, 512, 515, 519
BDAM files • 503, 505, 506, 507, 508, 509
breakpoint usage • 162
capabilities • 441
changes, and main storage • 162
common functions • 465
DB2 databases • 487
DL/I databases • 477
ending FILE session • 457
HELP • 460
initiating • 161, 443
LAST sub-function • 466

logging facility • 456
overview • 441
PF keys • 457
record identification • 466
saving records • 464
screen • 445, 464
copying • 464
initial • 445
scrolling • 458
Source Listing Breakpoint screen, access from • 161
temporary storage • 512, 515
temporary storage screen layout • 511
terminating • 457
transient data • 519
universal mode of data entry • 455
VSAM files • 465
work area • 443, 461, 462, 464
changing • 461
displaying • 464
dumping • 464
filling with character string • 462
file names, replacing • 245, 327, 359
files • 252, 324, 363
preventing updating • 252, 363
updating, preventing • 324
FOL option • 252, 319, 363, 370
FOL=ppt optiondynamically called programs • 268
FOL=PPT option • 268
format • 447, 449, 450, 451
character • 449
dump • 447
structured • 451
vertical • 450
FORWARD command • 41
From Terminal ID, changing from Source Listing Profile • 84
Front-End Programming Interface (FEPI), TIMEOUT values • 291
FUNC field, common functions • 465
Function Selection menu • 336, 337
FWD command, on Source Listing display • 57

G

generic specification • 238, 299, 338
global logging • 309, 371
GLOG keyword • 309, 371

H

HELP command • 41, 57
navigation • 41
on Source Listing display • 57
HELP facility • 31, 48, 143, 235, 334, 460, 543
automatic breakpoint usage • 143
browse capability • 48
CNTL • 334
FILE • 460
ITST • 235
printing • 543
transaction IDs, listed in • 48
using • 31
HOGAN Systems programs, monitoring • 274
host variables • 527
displaying • 527
overview • 527

I

ICT option • 319
INCL keyword • 304
indexed data names • 392, 414
displaying CORE command • 414
displaying CORE menu • 392
indirect addressing • 411
indirect commands • 187, 189, 196, 201, 205, 206, 207, 208, 209, 210, 220, 305, 366
accessing • 189
accessing from Source Listing • 189
CNTL command • 305
COBOL programs • 205
COBOL variable syntax • 209
examples • 187
listing • 201
PL/I programs • 206
PL/I variable syntax • 210
Resume menu access • 220
status report • 196
using • 187
using conditional expressions • 208
using data names • 205
using figurative constants • 207
using variables • 205
using with CNTL menus • 366
Indirect Commands screen, description • 189
INQ keyword • 308, 370
installation options • 83, 232
Source List BKPT field • 83

viewing online • 232
instruction counter • 319
interactive testing • 24
ITST command, on Source Listing display • 57
ITST transaction, accessing Primary Option Menu • 36

J

JUMP command, on Source Listing display • 57

K

Keep window functions • 148, 149, 151
KEP option • 320

L

LAST sub-function • 466
LE/370 condition handlers • 268
ABI=OFF option • 268
segmented monitoring • 268
LET option • 249, 321, 361
Linkage section • 73
LIST keyword • 308
load module, modifying • 249, 321, 361
LOC field • 458
LOG field, temporary storage • 511

M

main storage • 29, 148, 152, 156, 178, 249, 252, 318, 320, 321, 323, 325, 361, 363, 375, 376, 377, 379, 385, 391, 392, 397, 398, 399, 401, 404, 405, 407, 411, 412, 413
contents • 178
deleting program modules • 404
display formats • 376
displaying • 407
displaying at breakpoint • 399, 401
displaying at breakpoint (COBOL and PL/I) • 412, 413
displaying at breakpoint (COBOL) • 391, 392
displaying complex COBOL data names • 392
displaying in a Keep window • 320
displaying system-related areas • 401
displaying task-related areas • 405
displaying, from Source Listing Breakpoint screen • 152
dump format • 385
examining • 29
indirect addressing • 411
instructions that read • 318
keeping, in a Keep window • 148
limiting • 252, 323, 363
loading program modules • 404
modifying • 29, 249, 321, 361
modifying at breakpoint • 399, 401
modifying system-related areas • 405
modifying task-related areas • 405
modifying, example of • 397
modifying, from Source Listing Breakpoint screen • 152
MOVE command, preformatted • 156
overview • 375
protecting • 249, 325, 361
searching for data • 398, 405
structured display • 377, 379
Assembler and CICS • 379
COBOL and PL/I • 377
viewing in keep window • 148
Main Storage facility • 375
access • 375
capabilities • 375
Margin field • 75
MENU command • 37, 57
on Source Listing display • 57
to display breakpoint menu • 37
menus and displays • 35, 38, 39, 40, 41
breakpoint menus and submenus • 40
commands for Option field • 41
getting help • 38
option selection • 38
overview • 35
structure and navigation • 39
menusanddisplays • 38
PFkeydefinition • 38
modify registers • 77
modules (CNTL commands), executing • 311
MON option • 278, 322
description • 322
setting and removing, alternate methods • 278
MONITOR command • 50, 57
on Source Listing display • 57
on Source Listing screen • 50

monitoring • 49, 89, 90, 100, 108, 125, 131, 132, 224, 230, 235, 238, 239, 240, 241, 242, 243, 245, 247, 248, 249, 251, 252, 255, 256, 257, 259, 260, 264, 269, 270, 271, 272, 273, 274, 295, 299, 300, 302, 303, 304, 305, 307, 308, 309, 311, 313, 315, 316, 319, 325, 327, 330, 331, 334, 338, 339, 340, 341, 342, 346, 355, 357, 359, 360, 361, 362, 363, 367, 370, 371, 372, 374, 571
abending tasks • 307
all programs • 239
all users of a program • 302
automatic breakpoints • 252, 269, 272, 273, 309, 315, 363, 371
non-3270 terminal tasks • 273
CNTL command and menus • 295
COBOL II and COBOL/370 • 571
composite support • 255, 256, 257, 259, 260
conditional breakpoints • 132, 316, 346
data • 100
dynamically called programs • 571
from Source Listing facility • 49
generic specification • 238, 299, 338
global • 239, 302, 371
hierarchy rules • 239, 300, 339
initiation methods • 235, 334
limiting • 243, 245, 249, 252, 270, 330, 359, 361, 363
MSA programs under CA InterTest for CICS • 274
non-3270 terminal tasks • 273
non-terminal tasks • 272
options • 302, 303, 304, 311, 313
excluding • 304
removing • 302, 313
setting • 302, 303, 313
summary • 311
production system • 269, 271
programs, multiple • 90
protection options • 248, 251, 361, 362
removing • 240, 241, 302, 340
replacement options • 242, 245, 247, 359, 360
report, global • 308, 370
request breakpoints • 125, 325, 327, 355
resetting • 309, 370
resuming task execution • 304, 305, 372, 374
setting • 240, 302, 340
special options • 252, 363
Statement Trace options • 357
status after testing • 224
status report • 131, 240, 300, 308, 339, 340, 341, 346, 367
system-wide options • 239, 264, 302, 309, 371
terminal options • 240, 243, 245, 249, 252, 270, 330, 340, 359, 361, 363
UBP optionSee unconditional breakpoints • 108
unconditional breakpoints • 331, 342
User ID options • 89, 230, 270
using menus to remove • 241
utility functions • 370
wild branch • 252, 319, 363
Monitoring activitySee Monitoring status display • 224
Monitoring Command Builder • 335, 336, 337
how to use • 335
main menu • 336, 337
Monitoring menu • 227, 228, 229, 230
access • 229
functions • 228
options • 227
PF keys • 230
Monitoring Status display • 42, 43, 44, 45, 46, 224
accessing • 44
CNTL command entry • 224
exiting • 46
expanding and collapsing • 45
introduction • 43
refreshing the display • 46
removing monitoring, breakpoints, options • 45
using • 42
Monitoring submenus • 234
monitoring table • 302, 308, 340
automatic creation of entries • 302, 340
displaying • 308
MUS option • 252, 270, 363
MXR option • 252, 363
MXS option • 252, 363

N

Nested program field • 67, 147
New Copy option • 225, 244
functions • 225
uses • 244
NEW keyword • 244, 309, 370
New Program Copy • 105, 225
execute • 225
transfers breakpoints • 105
no file updating option • 252, 324, 363

NOM option • 323
Non-symbolic Breakpoint screen • 549, 550, 552, 555, 556
NRB option • 324
NUP option • 252, 324, 363

O

OFF keyword • 302
Offset location • 73
ON keyword • 302
Option field, ITST menus, commands supported • 41
override automatic breakpoint • 145
OVR option • 324

P

passwords, files and databases • 456
PBP facility • 288, 290
PF keys • 38, 66, 142, 171, 173, 181, 230, 340, 387, 389, 457, 493
  CNTL • 340
  CORE (menus) • 389
  CORE (structured display) • 387, 389
  detailed breakpoint • 181
  FILE • 457
  menus • 38
  Monitoring menu • 230
  Source Backtrace Summary • 171
  Source Listing Backtrace • 173
  Source Listing Breakpoint • 142
  Source listing display • 66
  SQL records in FILE • 493
Plus signs • 167
post-processor programs, recompiling with • 225
Primary Option Menu • 36, 37, 38, 163
  access from Breakpoint menu • 163
  accessing • 36
  accessing from a breakpoint • 37
  PF key definitions • 38
PRO option • 249, 325, 361
production system • 28, 269, 270, 271
  automatic breakpoints • 269, 270
  CICS • 28
  limiting monitoring • 270
  monitoring • 269, 271
  monitoring at one terminal • 270
  monitoring for a single user • 270
PROFILE command • 57, 76
  accessing • 76
  on Source Listing display • 57
program execution • 174, 213, 215, 218
  jump to another location • 215
  navigating • 174
  options, at a breakpoint • 213
  resume menu • 218
Program Monitoring menu • 108, 244
  new copy option • 244
  setting an unconditional breakpoint • 108
program names • 245, 327, 359
  replacing • 245, 327, 359
programmed breakpoints • 288, 290
  PL/I programs • 288
  removing • 290
programs • 49, 75, 223, 225, 238, 299, 308, 338, 370, 545
  adjusting the margins • 75
  generic specification • 238, 299, 338
  non-symbolic • 545
  recompiling after testing • 225
  status after testing • 223
  Symbolic file, listing of • 308, 370
  viewing online • 49
protection options • 248, 251, 361, 362
  removing • 248, 251, 361, 362
  setting • 251, 361, 362
PURGE keyword • 309, 371

Q

qualified data names • 392, 414
  displaying CORE command • 414
  displaying CORE menu • 392
qualified name, COBOL nested programs • 91

R

RBP option • 325, 327
RCT - See Resource Control Table • 537
RCT keyword • 537
RDIIN keyword • 537
read buffer, preventing • 324
reentrancy check • 252, 327, 363
registers • 77
  Amode field • 77
  Condition Code field • 77
  display • 77
  ExecKey field • 77
  modify • 77
  TransIsolate field • 77

view the area pointed to • 77
replacement options • 242, 245, 327, 328, 359
removing • 242, 245
removing, CNTL menus • 359
setting • 242, 245, 327, 328
setting, CNTL menus • 359
Replacement Options screen • 245, 359
reports • 131, 300, 308, 339, 341, 346, 367, 370
global • 370
status • 131, 300, 308, 339, 341, 346, 367
request breakpoints • 27, 125, 133, 325, 327, 355, 526
defined • 27
for SQL requests • 526
removing • 125, 325, 327, 355
removing from CNTL • 355
removing from menus • 133
setting • 125, 325, 327, 355
Source Listing facility • 125
setting from CNTL • 355
setting within loop • 325
terminal options • 325
Resource Control Table • 498, 537
displaying • 537
viewing • 498
RESTART keyword • 301
Resume menu • 218, 219, 220, 221, 222
option 1 Next Instruction • 218
option 2 Label • 219
option 3 Statement • 219
option 4 Offset • 220
option 5 Indirect Commands • 220
option 6 Next EXEC/CALL • 221
option 7 nnn Steps • 221
option 8 nnn Machine Instructions • 222
resuming task execution • 304, 305, 372, 374
RFC option • 243, 245, 327, 359
RNT option • 252, 327, 363
RPC option • 243, 245, 327, 359
RTD option • 245, 328, 359
RTS option • 245, 328, 359
Run location • 84
different terminal • 84
no terminal • 84

S

scroll amount • 80
HALF • 80
PAGE • 80
setting • 80
STOP • 80
Search field • 66
searching for data, Source Listing facility • 66
segmented monitoring options • 275, 276, 277, 278, 280, 283, 284, 285
enabling • 277
installation requirements • 277
MON (monitor on) option, defined • 276
MONOM option, online status • 277
NOM (monitor off) option, defined • 276
password protection for • 277
setting and removing • 278, 280, 283, 284, 285
using CNTL menus • 280, 283
using command syntax • 283, 284, 285
using Source Listing • 278
used with HOGAN Systems programs • 275, 277
used with MSA programs • 275, 277
segmented monitoring, definition • 274
SELECT command • 490, 494
single-stepping • 215, 221, 372
SLB option • 252, 329, 363
Source List BKPT= field, default value • 186
Source Listing Backtrace • 167, 173
PF keys • 173
when to use • 167
Source Listing Breakpoint • 83
Source List BKPT= OFF • 83
Source List BKPT= ON • 83
Source Listing Breakpoint screen • 136, 137, 142, 143, 148, 152, 161, 252, 329, 363, 375
accessing/modifying databases • 161
accessing/modifying files • 161
automatic breakpoint example • 136
automatic breakpoint, help for • 143
breakpoint letters • 136
commands supported • 137
CORE, using • 375
displaying • 329
examining • 136
Keep window display • 148
main storage, display functions • 152
PF keys • 142
requesting • 252, 363
Source Listing display • 54, 56, 57, 66, 73, 75
adjusting the margins • 75
areas, defined • 54
breakpoint letter indicators • 54

Command line usage • 57
commands supported • 57
field definitions • 54
Option # field • 56
PF key definitions • 66
scrolling function • 73
using the Search field • 66
Source Listing facility • 32, 47, 49, 51, 73, 89, 90, 101, 105, 113, 125
access to • 32
exiting • 51, 101
initiating • 49, 73
monitoring multiple programs • 90
overview • 47
preparing your programs • 47
profile options • 73
program execution • 101
setting breakpoints • 49, 51
setting conditional breakpoints • 113
setting monitoring • 49
setting monitoring monitoring • 89
SLB option • 89
setting request breakpoints • 125
setting unconditionabreakpoints • 105
Source Listing Profile • 73, 76, 77, 80, 81, 82, 83, 84, 86
auto-stepping • 82
BKPT terminal ID • 84
default settings • 76
display registers • 77
display titles • 77
From terminal ID • 84
options • 76
PF7/8 Amount field • 73
scroll amount • 80
set the User ID • 86
setting scroll amount • 73
source list BKPT • 83
step amount • 81
Source menu • 51
Source Selection List • 51
accessing • 51
using Masking characters • 51
special options • 252, 363
removing • 252, 363
setting • 252, 363
User ID • 252
SQL • 526, 528
requests • 526
return codes • 528
SQL/DS programs • 526, 527, 528, 532
advanced debugging • 532
host variables, inspecting • 527
SQL requests, breakpoints for • 526
SQL return codes • 528
SQLCA keyword • 531
SQLCA, displaying • 531
SQLRCODE keyword • 530, 536
SSA formats • 480
START keyword • 68, 70, 71, 72, 301
Statement Trace facility • 158
enabling tracing and data monitoring • 158
overview • 158
Statement Trace options • 242, 243, 244, 357, 359
removing • 244, 357, 359
setting • 357
specifying • 243
TRC • 357
when to use • 242
Statement Trace Options screen • 243
statement tracing • 100
STATUS ALL command, on Source Listing display • 57
STATUS command, on Source Listing display • 57
Status display • 41, 131, 241, 300, 308, 339, 340, 341, 346, 367, 368
at source listing breakpoint • 131
commands supported • 41
determine status • 367
example • 368
hierarchy rules • 339
monitored entries • 300
monitoring report • 308
remove monitoring • 341
remove unconditional breakpoint • 346
removing monitoring • 241
set/remove monitoring • 340
Status/Maintenance menu, displaying global options • 232
step amount, setting • 81
stepping, automatic • 82
storage pointer • 409, 410
storage protection • 249, 332, 361
bypassing • 249, 361
removing from shared areas • 332
Structured Query Language (SQL) • 488
subscripted data names • 392, 414
displaying CORE command • 414
displaying CORE menu • 392

SVC instructions, permitting • 249
SYM keyword • 308, 370
Symbolic file • 308, 370
  programs, listing of • 308, 370
Symbolic Version List • 93, 94
  assembler programs • 94
  COBOL programs • 93
  PL/I programs • 93
  using • 94
Synchronized processing • 92
  maintaining • 92
  Symbolic Version List • 92
  warning message • 92
System-Related Areas menu • 401, 404
system-wide monitoring, using menus • 239
system-wide options • 264, 302, 309, 371

T

Task-Related Areas menu • 405
tasks • 272, 304, 305, 307, 309, 371, 372, 374, 405
  abending • 307, 372
  CORE menus • 405
  non-terminal • 272
  purging • 309, 371
  resuming execution • 304, 305, 372, 374
temporary storage • 245, 328, 359, 511, 512, 514, 515
  adding a queue record • 514
  FILE functions for • 512
  FILE screen layout • 511
  purging a queue • 515
  replacing IDs • 245, 328, 359
  viewing a queue record • 512
TER option • 330, 332
Terminal Monitoring menu, setting an unconditional breakpoint • 108
terminals • 238, 263, 273, 299, 330, 332, 338
  changing IDs • 330, 332
  disconnecting active task • 263
  generic specification • 238, 299, 338
  non-3270 terminal • 273
test cycle activities • 47, 49
  preliminaries • 47
  starting a session • 49
test system • 28
titles, display • 77
TON option • 243, 245, 249, 252, 270, 330, 359, 361, 363
TOP command, scrolling an Options list • 237
TRACE command • 158
Trace Table • 566, 568
  CA InterTest for CICS entries • 566, 568
tracing statements • 100
transaction dump • 304, 307
Transaction IDs • 48
  for CA InterTest for CICS • 48
  for CA SymDump for CICS • 48
Transaction Monitoring menu, setting an unconditional breakpoint • 108
transactions, generic specification • 238, 299, 338
transient data • 519
  adding a record • 519
  retrieving a record • 519
transient data queues, replacing • 245, 328, 359
TRC option • 330, 357

U

UBP option • 331
unconditional breakpoints • 27, 105, 108, 111, 130, 131, 331, 342, 346
  defined • 27
  indirect command option • 111
  removing • 130, 131, 342, 346
  set at all labels, paragraphs, or procedures • 105
  set at labels, paragraphs, or procedure names • 105
  set from Cross-Reference section • 105
  setting • 331, 342
  setting from menus • 108
  setting from Source Listing • 105
  setting within loop • 111, 331
  specifying options • 111
  terminal options • 111, 331
  user ID option • 111
USE option • 381
User ID • 86, 112, 249, 332
  changing from Source Listing • 86
  example, setting breakpoints from menus • 112
  option to monitoring, command syntax • 332
  protection options • 249
  Source Listing Profile • 86
USH option • 332
USR keyword, USR=specific • 302
USR option • 270, 332
  in production • 270
  limit monitoring • 332

utility functions, usage • 370

V

variable length data, displaying CORE command • 414
variable-change breakpoints • 118, 349
  setting • 349
  setting from menus • 118
VSAM files • 442, 465, 467, 469, 470, 471, 472, 473, 474, 475, 476, 477
  browsing • 470
  FILE functions for • 469
  FILE screen • 467
  mass insertion • 474
  record • 465, 469, 472, 473, 475, 476, 477
    adding • 473
    copying • 475
    deleting • 476
    deleting large-key • 477
    identifying • 465
    updating • 472
    viewing • 469
  remote files and temporary storage support • 442
  searching for data string • 471

W

What's new (8), on Primary Option Menu • 37
wild branch • 252, 319, 363
Working-storage • 73
Wrong results • 101