

CA InterTest™ and CA SymDump® - 11.0

Using Batch Tools

Date: 04-Jun-2018



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 © 2018 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Table of Contents

Command Syntax	13
----------------------	----

Batch Debugging	14
-----------------------	----

Debug Optimized Applications	15
------------------------------------	----

COBOL Statement Numbers	16
-------------------------------	----

Assembler Offsets	16
-------------------------	----

Panels and Functions	16
----------------------------	----

Panel Display Formats	16
-----------------------------	----

Program Access Keys	17
---------------------------	----

Program Access Keys	17
---------------------------	----

Primary Option Menu	17
---------------------------	----

Debugging Panels	19
------------------------	----

Overview of Test Panels	20
-------------------------------	----

Execution Control Panel	21
-------------------------------	----

Allocations Fields	21
--------------------------	----

Execution Overrides	21
---------------------------	----

Monitor Control Panel	22
-----------------------------	----

Monitored Programs	22
--------------------------	----

Symbolic Files	22
----------------------	----

Dynamic Symbolic Support	23
--------------------------------	----

Dynamic Symbolic Support Return Codes	23
---	----

Intercept Panel	24
-----------------------	----

Types of Intercepts	25
---------------------------	----

Breakpoint Panels	26
-------------------------	----

Breakpoint Panel	27
------------------------	----

Breakpoint Status Panel	27
-------------------------------	----

When (or COND) Panel	28
----------------------------	----

When Conditions Display Panel	30
-------------------------------------	----

Equate Panel	30
--------------------	----

Equate Display Panel	31
----------------------------	----

Data Display Panels	31
---------------------------	----

Display Panel	31
---------------------	----

Linkage Panel	33
---------------------	----

Records Panel	34
---------------------	----

File Status Panel	34
-------------------------	----

Program Trace Display Panel	35
Core Option	36
DISPLAY	36
FIND Command	37
FX (Find Hex Data)	37
POP	38
PUSH	38
Map Option	38
Debugging Commands	40
Display Commands	40
COLUMN Command	41
FIND Command	41
FP Command (Find Paragraph)	42
FS Command (Find Statement)	42
KDOWN Command (Scroll Keep Window Down)	43
KSIZE (or KS) Command (Keep Window Size)	44
KUP Command (Scroll Keep Window Up)	44
LOCATE Command	45
.label Command	46
Line Commands	46
A or U Command (Set Unconditional "Before" Breakpoints)	49
C or W Command (Set Conditional Breakpoint)	50
D Command (Display Working Storage)	51
G Command (Go from Line)	51
H Command (Display Hex Data)	52
I Command (Display Full Frequency Count Information)	53
L or K Command (Display Data Item in a Keep Window)	54
O or X Command (Turn Off Breakpoints)	55
P Command (Print Display)	57
R Command (Reset Data Item Display)	57
S Command (Skip this Statement)	58
V Command (Set Variable Change Breakpoint)	59
) Command (Set Unconditional "After" Breakpoints)	60
+ (plus) Command (Increment a Subscript)	63
- (minus) Command (Decrement a Subscript)	64
< nnnn (Scroll Left) Command	65
> nnnn (Scroll Right) Command	66
Control Commands	67
ADVANCE Command (Follow Trace Pointers Forward)	70
AT (or UNCOND) Command (Set an Unconditional Breakpoint)	71
CORE Command (Display Memory)	73
COUNTS Command (Control Frequency Counting)	73

CS Command (Current Statement)	74
DATAMON (Enable Data Monitoring)	74
DDALLOC Command (Allocate a DD)	75
DDFREE Command (Free a DD)	75
DDQ Command (Query a DD)	75
DISPLAY Command (Format Data or the COBOL Working Storage Section)	76
DROP Command (DROP Symbolic Name)	77
DROPUSE Command (Remove a Using)	77
DUMP Command (Terminate Testing with a Dump)	78
EQUATE Command (Specify Symbolic Name)	79
EXSUM Command (Display Execution Summary)	80
FILES Command (Display COBOL FD Status)	81
FM Command (Invoke CA File Master Plus)	81
FREQ Command (Controls FREQ Counter)	81
GO Command (Continue Test Session)	82
INCLUDE Command (Execute Application Commands)	83
LDA (or AUTOKEEP) Command (List Data Automatic)	84
LDI (or KEEP) Command (List Data Item)	85
LDX (or KEEPX) Command (List Data Item in Hex)	87
LEQ Command (List Equated Names)	89
LINKAGE Command (Format the COBOL Linkage Section)	89
LISTAT Command (List Breakpoints)	90
LISTBP Command (List All Breakpoints)	90
LISTLBL Command (List Labels)	90
LISTUSE Command (List Usings)	90
LISTWHEN Command (List WHEN Conditions)	91
MAP Command (Region MAP Display)	91
MLOG Command (Record Debugging Session)	91
NEXT Command (Executing Verbs)	92
NOFREQ Command (Remove Frequency Counters)	93
OFF (or OFFU) Command (Remove Unconditional Breakpoints)	93
OFFWN (or OFFC) Command (Remove Conditional Breakpoints)	95
POINT Command (Direct PREV and ADVANCE Pointer)	95
PREV Command (Trace in Reverse)	96
PS Command (Print Stream)	97
QUALIFY Command (Set Current Program Id)	97
QUIT Command (Terminate Session)	98
RDI or REMOVE Command (Reset Data Item)	98
RECORDS Command (Show COBOL Record Formats)	99
REFRESH Command	100
REGS Command	100
RESET Command	100
RESTART Command	101

RUN Command (Resume Execution)	101
SDWA Command (System Diagnostic Work Area)	101
SET Command (Change Data Item Value)	102
SKIP Command (Skip a Statement)	103
SLOW (or AUTOSTEP) Command (Slowly Resume Program Execution)	105
SNAP Command (Produce a SNAP Report)	105
SPEED Command (Accelerate Program Testing)	106
STEP Command (Set Count for GO Command)	106
SUSPEND Command (Suspend a Batch Link Session)	107
TRACE SOURCE Command (Trace with Source Code Displayed)	108
TRACE Command (Controls Program Trace Entries)	108
TRP Command (Trace Paragraphs)	109
USING Command (Assign a Register or Address to a DSECT or CSECT)	109
WHEN (or COND) Command (Set Conditional Breakpoints)	109
* Command (Add Comments in Session Log)	112
/clist Command (Execute CAMRCMD)	112
Report Commands	113
HISTOGRAM Command (Graph Execution Frequencies)	113
XSUM Command (Execution Summary Report)	115
Session Log Facility (Review Debugging Session)	117
Testing Procedures	118
Guidelines for Novice Users	119
Preparation for Test Execution	119
Execute a Program Until an Error Occurs	120
Stop a Looping Program	121
Trace a Program in Reverse	121
Stop a Program at a Particular Statement	121
Stop at a Statement and Step by Verb	122
Stop a Program When a Subscript Overflows	123
Test IMS Applications	124
IMS Batch Program Testing	124
Batch Procedures for BMP Testing	126
Test Procedure Using BTS	130
BTS Interactions	131
Test COBOL Programs Which Run Under an ADF Shell	132
Test IBM DB2 Applications	133
Test Data Generation	134
Test Under CA Roscoe	135
Allocations Facility for ISPF	136
The Allocations Main Menu	137
Option 1 ALIB (Allocations Library) Functions	137
Select an ALIB Member from the Allocations Library Panel	137

Select an ALIB Member from the ALIB Member Selection List	138
Edit ALIB Allocations	138
Convert ALIB to CLIST (Export ALIB)	143
Convert JCL to ALIB (Import JCL)	144
Option 2 The JCL Library/Allocations Functions	145
Specify the JCL to Convert	145
Use the JCL Member Selection List	146
The Conversion Process	147
Option 3 Current Allocations	149
Option 4 JCL Conversion/Allocation Options	150
JCL Conversion for CA Roscoe	151
Access the JCL Conversion Facility	152
Specify the JCL Member for Conversion	152
Identify Where the JCL Member Is Stored	152
Identify the PROCLIBs	153
Display the JCL Member	153
Select the JCL Member from a List	153
Convert the JCL Member	154
Review and Edit the Converted JCL	154
Execute the Converted JCL	155
Convert JCL to ALIB in Batch	155
Batch Link Facility	155
CA InterTest Batch JCL Requirements	156
Debug a Batch Application	157
Use the Batch Link Menus	158
Batch Link JCL Conversion Panel	158
Batch Link Selection Panel	162
Batch Link JCL Edit Panel	164
DB2 and IMS Schedule Menu	167
Batch Link Special Considerations	173
DB2 Considerations	173
Debug Your DB2 Stored Procedures	174
Debug Your Online IMS Programs	174
Use Batch Link Schedule to Enhance your DB2 SP or IMS/DC Debugging Experience	174
Import and Export Schedule Files	176
CA InterTest Batch Utilities	177
Display the Contents of the Symbolic File	178
Add a Program to the PROTSYM File	178
Exclude Programs from Auto-monitoring	179
Work With Recorded Debug Sessions	179
DDnames and Sample CLISTs	181
DDnames Required for Proper Execution	181

Optional DDnames	182
Sample CLISTs to Invoke the Application	183
Other CLISTs	183
Batch Abend Analysis	185
Components	185
Using Symbolic Support	186
Symbolic Support Features	186
Use Existing CSL Files	187
Access Symbolic Information at Execution Time	187
Add Symbolic Information Using the Viewer	188
Reporting	189
Hierarchy of Information	189
Usage Considerations	189
Reporting Options	190
Standard Options	198
Formatting Options	198
VSAM Control Block Options	198
Data Management Control Block Options	199
Coding Options	199
Override Defaults	199
CA SymDump Batch JCL Requirements	200
Repository Requirements	200
Report File Requirements	200
Option File Requirements	200
Symbolic File Requirements	200
Suppress Abend Reports	201
Reports	201
Abend Report	202
Snap Report	256
CAIOPTS File Processing Report	258
Report Summary	258
Merged Versus Dumped Data Displays	259
Merged Display	259
Dumped Display	262
CAIPRINT Repository Viewer	263
Report Index	264
Select a Repository	264
Report Index Fields	265
Primary Commands	266
Line Commands	268

Delete a Report	268
Lock and Unlock Reports	268
Use the Electronic Notepad	269
Modify Formatting Options	269
Print a Report	271
Select a Report for Viewing	273
Report Tree	273
Report Tree Fields	274
Primary Commands	275
Line Commands	275
Expand and Collapse the Tree View	276
View a Report	277
Report View Fields	277
Advanced Techniques	277
Set PF Keys	278
Use the Keep Window	279
Use the Electronic Notepad	280
Use TAG and LOCATE	282
Symbolic Utilities	283
List the Contents of a Symbolic File	284
Delete a Member from a Symbolic File	285
Print a Program Listing from a Symbolic File	286
View a Program Listing from a Symbolic File	288
Add a Listing to a PROTSYM File	289
List Globally Defined Symbolic Files	291
List Supplemental Symbolic Files	291
View Dynamic Symbolic Support Options	292
Override Dynamic Symbolic Support Options	293
Repository Viewer Commands	294
FIND	294
FM	296
HELP	296
KEEP	297
LOCATE	298
NOTES	298
OPTIONS	299
PRINT	299
PROFILE	301
REFRESH	303
RFIND	303
SETINDEX	304
SORT	305
SYM	305

TAG	306
VIEW	306
CA SymDump Batch Utilities	307
Management Reporting System (MRS)	307
JCL	308
Reports	308
Usage Considerations	312
Options Summary Report	313
JCL	313
Report	313
CSL Summary Report	314
JCL	315
Report	315
CAIPRINT Repository Utility	316
JCL	316
Commands	317
Dynamic Symbolic Support	320
Dynamic Symbolic Support Return Codes	321
C1DEFLT5 Consideration	321
Troubleshooting	322
Collect Diagnostic Data	322
Execution-Time Problems	322
Formatting Problems	322
Symbolic Postprocessors	323
Interpret Diagnostic Data	323
Print a Symbolic File Member for Diagnostic Purposes	324
Use the Viewer	324
Use the Batch Utilities	324
Copy Repository Data for Diagnostic Purposes	324
Create a Temporary VSAM Repository	325
Copy the Report to a Temporary VSAM Repository	325
Create a Sequential Data Set from a VSAM Repository	325

Using Batch Tools

CA InterTest Batch is a comprehensive interactive tool for testing and debugging batch, DB2 stored procedures, and IMS/DC applications through a TSO/ISPF or CA Roscoe Interactive Environment (CA Roscoe) interface. Programmers use interactive, online facilities with the actual source listing for debugging, rather than dumps or reports, so you can resolve many errors in a single test session. The application intercepts all application abend conditions and identifies the source code statement that caused each error. Interactive debugging functions include the ability to set breakpoints, trace program execution, place data items in a keep window on the source listing screen, examine and modify the values of data items, and find a data string in working storage. A user-friendly, menu-driven allocation and JCL conversion facility simplifies the task of allocating program files for foreground execution or preparing JCL for batch execution.

CA SymDump Batch provides abend diagnostic information for batch abends in test and production environments. Its formatted reports make it easy for programmers to resolve abends without having to decipher system dumps. CA SymDump Batch supports all languages, with special symbolic support for COBOL, PL/I, and Assembler. It also includes detailed database information for abending IMS, DB2, and CA IDMS/DB applications. CA SymDump Batch is automatically available to every user once it is installed.

Command Syntax

The following notation conventions are used throughout the documentation to illustrate application commands:

- Uppercase letters and special characters are displayed as usual.
- Lowercase *italic* letters indicate the generic description of the value of a variable parameter.
- Brackets (< >) enclose optional values.
- Vertical bars (|) separate alternatives.
- Ellipses (...) specify that a parameter can be repeated.
- Default values are underlined.

Batch Debugging

This section describes how to use CA InterTest Batch to test a program using application batch commands and panels. The application is supported with HELP panels and an online tutorial, which describe the application options and commands. You can use this product to test the following programs:

- Assembler programs that are assembled with Assembler F, Assembler H, or the High Level Assembler
- COBOL programs that are compiled with any COBOL compiler supported by IBM
- PL/I programs that are compiled with any PL/I compiler supported by IBM

The application gives programmers complete access to the program as it is executing. You can stop the program and the Intercept panel displayed at user-specified intercept points. From the Intercept panel, you can browse the program listing in full-screen mode, using program function keys (PFKeys) to scroll backward or forward through the program. You can examine and repair problem areas without ending the test session, or issue commands to set new intercept points and resume testing. Each application panel is supported with a HELP panel. An online tutorial describes the product system.

During a test session, you can perform the following actions:

- Stop the test at any executable statement in the program.
- View the program listing while the test session is stopped.
- Trace the program execution and browse the trace table entries.
- Display and alter data-item values.
- Step through the program in increments of any number of statements.
- Identify the number of times each statement in the program is executed.
- Step through the program execution in reverse order.
- Access the other options such as Core.

The application runs under IBM's ISPF and CA Roscoe Interactive Environment (CA Roscoe) under ETSO. Programmers familiar with these environments need little or no training to use this product.

During a test session using the ISPF dialog manager, programmers can use the following features to test efficiently:

- Split the screen to access other ISPF options, such as EDIT.
- Print to the ISPF listing data set.
- Enter ISPF or TSO commands from any panel.

During a test session using CA Roscoe, programmers can perform the following features:

- Split the screen to perform other CA InterTest Batch or CA Roscoe functions.
- Suspend the application session to perform other CA Roscoe functions.
- Print to an application data set, which can be allocated to the AWS or any MVS data set.
- Use the CA Roscoe print facility.

Debug Optimized Applications

You can use CA InterTest Batch to debug programs that were optimized by the COBOL compiler OPTIMIZE option or by CA Optimizer or CA Optimizer/II. However, debugging these programs can sometimes result in unexpected behavior.

During optimization, a compiler often relocates individual instructions, statements, or even entire paragraphs. If instructions are moved to another statement or if statements in a paragraph are moved to another paragraph, the resulting object program and listing might 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 might be times when the breakpoint intercept does not occur, or when the wrong sequence of statements appears to be executed while single-stepping. There might 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. When they occur, they indicate that the debugger cannot accurately identify exactly which object code corresponds to which source statement, or which statement is contained within which paragraph.

The application 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. Inaccuracies can include the following instances:

- Incorrect execution when using the SKIP, GO stmt# or CS stmt# commands
- Failure to stop at a breakpoint at a paragraph label or statement
- Unexpected or out of sequence highlighting of statements when single-stepping

Application abends can result from the use of the SKIP, GO stmt#, or CS stmt# commands because the optimized object code might have register requirements that do not support changes to the flow of control. Avoid these commands when debugging an optimized program.

For the best debugging results, avoid using optimization whenever possible in your testing environment.

COBOL Statement Numbers

Some commands require a statement number in the command syntax. For COBOL programs, these statement numbers are generated by the COBOL compiler. The compiler-generated statement numbers are the ones farthest to the left in the compilation listings and in application displays. Use these numbers in all commands that require a statement number.

Assembler Offsets

Use the statement number in an Assembler program to set breakpoints, as in COBOL programs, or by using the offset. When using an offset, you must prefix the offset with the + sign.

For example, enter AT +78 in the Command area of the Intercept panel to set a breakpoint at offset 78. You can also enter S in the Option field and +78 in the Statement field on the Breakpoint panel.

Panels and Functions

This article describes how panels are displayed, the options available, and how to understand the syntax in product documentation.

Panel Display Formats

The application uses full screen displays to prompt you for option selection and data entry, and to display source data, listings, and other information.

The first two lines are the header lines; they have a common format for all displays.

Line one of the application panel contains the panel title and short message area. The title line identifies the reason for the panel being displayed, and, where appropriate, data set information. The short message area displays the following information:

- Current line number (in BROWSE)
- Notice of successful completion of a processing function
- Notice of an error condition (accompanied by an audible alarm, if one is installed on the terminal)
- A frequency count in response to an Info line command

Line two, the prompt or input area, is used to enter an option, selection, or command.

The remainder of the screen contains a list of options, input fields and prompts, or scrollable data. You can scroll in any application panel containing a SCROLL field in line two. The application recognizes PAGE (P), HALF (H), MAX (M), or four numeric digits. The default is PAGE. Enter a numeric value in the SCROLL field to control the number of lines scrolled when you press the SCROLL key.

A scroll field value of cursor (CSR) is supported under CA InterTest Batch ISPF and CA Roscoe.

The application does not display ISPF long messages.

Program Access Keys

For ISPF users, the application uses the PF keys as set from the ISPF option. For CA Roscoe users, the application uses the PF keys as set from the CA InterTest Batch Primary Option Menu.

Program Access Keys

The program access keys are defined as follows:

- **ATTENTION (PA1)**

Interrupts the program that is currently executing and displays the ATTENTION INTERCEPT panel at the next executable statement in a monitored program. If the program is in a loop, you might need to press RESET before pressing PA1.



Note: The ATTENTION key can be either the PA1 key or the ATTN key, depending on the terminal type and how it is attached to the system.

- **RESHOW (PA2)**

Redisplays the contents of the screen. Use this key if, for example, the CLEAR key was pressed accidentally.

Primary Option Menu

The following list shows the features available from the Primary Option Menu. Availability of individual options depends on whether you are using ISPF or CA Roscoe.

- **Option 1**

- **Foreground**

Use this option to set up and control a test execution of a program.

- **Option 2**

- **Core**

Use this option to display virtual memory in hexadecimal and character format. Use SCROLL keys and commands to locate and find data in virtual storage.

- **Option 3**

- Allocation**

- ISPF users can select this option to allocate ddnames and allow conversion of JCL into a CLIST or ALIB.

- **Option 4**

- Map**

- This option provides several address space-related displays that are most often used to debug complex system-related problems.

- **Option 5**

- Batch**

- Use this option to set up and test a program to be executed in Batch.

- **Option K**

- Keys**

- Use this option to set up and control a test execution of a program.

- **Option K**

- Keys**

CA Roscoe users can set their PF keys for CA InterTest Batch. Type over the default assignments on the PF Key Definition panel and press Enter to change the PF keys. The default key assignments are as follows:

- **PF1**

- Help

- **PF2**

- Split

- **PF3**

- End

- **PF4**

- Return

- **PF5**

- Find

- **PF6**

- Go

- **PF7**

- Up

- **PF8**

- Down

- **PF9**

- Swap

- **PF10**
Left
- **PF11**
Right
- **PF12**
Print

Valid PF key commands are HELP, SPLIT, END, RETURN, FIND, GO, UP, DOWN, SWAP, LEFT, RIGHT, and PRINT.

- **Option U**
Utilities

Use this option to display and update the contents of your PROTSYM file. You can also define which programs to exclude from auto-monitoring.

- **Option X**
Exit

Use this option to terminate the application. The ISPF End key (typically PF3) or CA Roscoe End Key (typically PF4) also terminate the application from the Main Menu.

- **ISPF-Help**
Help

Display the Tutorial Table of Contents from the Primary Option Menu. The tutorial summarizes the features and commands. You can page through the tutorial or select a subject from the Tutorial Table of Contents.

The Primary Option Menu for CA Roscoe users is displayed when you select the CA InterTest Batch Option on the CA Roscoe menu or enter the following commands, where xxx is the prefix of the library where the application is installed. Omit this prefix if the application is installed in a common execution library.

```
[xxx.] IBALLOD
[xxx.] IBRUN
```

Debugging Panels

Foreground, Option 1 on the Primary Option Menu, includes the following panels:

- Panels to identify the programs to be tested, and their associated symbolic files
- A panel to view the program listing and control the test execution
- Panels to display and alter program storage
- Panels to display trace, file, and other related information



More information:

[Debugging Commands \(see page 40\)](#)

[Testing Procedures \(see page 118\)](#)

[Basic Foreground Demo Session \(https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session\)](https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session)

[Basic Batch Link Demo \(https://docops.ca.com/display/CAITSD11/Basic+Batch+Link+Demo\)](https://docops.ca.com/display/CAITSD11/Basic+Batch+Link+Demo)

[Advanced Demo Session \(https://docops.ca.com/display/CAITSD11/Advanced+Demo+Session\)](https://docops.ca.com/display/CAITSD11/Advanced+Demo+Session) .

In addition, novice users can gain online experience with the application by performing the basic demonstration sessions in the sections

Overview of Test Panels

The testing process begins by selecting Option 1, Foreground, on the main menu. The first two panels are called the Execution Control panel and the Monitor Control panel. Once you enter data on the Execution Control and Monitor Control panels, execution of the test program begins.

The following list describes a list of test-related panels:

- **Breakpoint Control**
That panel allows you to set breakpoints (places where the application should interrupt) by statement number or paragraph name.
- **Breakpoint Display**
This panel lists the breakpoints set by statement numbers or paragraph names.
- **When Control**
This panel allows you to set breakpoints based on variable contents or any time a variable changes.
- **When Condition Display**
This panel displays the when conditions set by either the line mode COND command or the When Control panel.
- **Equate**
This panel allows data items to be equated to symbolic names. You can then refer to the variable using the data item or the symbolic name.
- **Equate Display**
This panel displays the equate set by the line mode EQUATE command or the Equate panel
- **Data Display**
This panel shows a formatted display of program data.
- **Linkage Display**
This panel shows a formatted display of the LINKAGE SECTION in COBOL format. This display is only available for COBOL programs.

- **Records Display**

This panel shows a formatted display of the records defined under the FILE SECTION in COBOL format. This display is only available for COBOL programs.

- **File Status Panel**

This panel shows the allocation for each FILE DESCRIPTION by ddname and dsname, and the DCB information whether the file is open or closed. If it is open, it indicates for INPUT, OUTPUT or INOUT. This display is only available for COBOL programs.

- **Program Trace Display**

This panel displays program statements in the order of their execution.

- **Debug Session Display**

This panel displays a list of saved debug sessions that are stored in the INT1CLIB file and can be displayed, maintained, and loaded into an active debugging session. Use the MLOG command to launch this screen.

Access the other options during the test session by using the split-screen feature and selecting another option from the Primary Option Menu.

Execution Control Panel

Use the Execution Control panel to specify the main program, execution parameters, and allocations required to run the application.

Allocations Fields

Use the ALIB Dsname and Member fields to specify which existing ALIB member to use to perform the necessary allocations, specify the parameters to be passed to the main program, or to identify the main program.

If the ALIB contains multiple job steps, indicate which job step to use in the EXEC Job Step and Proc Step fields. Specifying an ALIB is not required.

Execution Overrides

Use the Execution Overrides fields to override any fields that were obtained through the ALIB, if any were specified. The fields are as follows:

- **PGM**

Indicates and overrides the main program to execute in this debug session.

- **PARM**

Specifies and overrides the execution parameter to pass to the main program.



Note: PARM is mutually exclusive with the Number of Linkage Parameters field.

- **Number of Linkage Parameters**

Specifies the number of linkage parameters expected by the main program when there is more than one. (For a COBOL program, this number is equivalent to the number of Linkage Section data items defined on the PROCEDURE DIVISION USING statement.) You can initialize and update the parameter values using the SET and LINKAGE commands. It also allows you to debug a subprogram as a main program.

- **Task Libraries**

Specifies and overrides the STEPLIB (or JOBLIB if one was specified in the ALIB) required to execute the application. Type the fully qualified data set name in quotes or indicate an allocated ddname in parentheses.



Note: For customers with multiple CA Endevor SCM C1DEFLT5:

If you want to use the dynamic symbolic support feature and you have chosen to run the debugging session as if you have a single C1DEFLT5 (for example, if you have not defined your CA Endevor SCM site IDs in IN25SITE table), define the fully qualified data set name that contains the C1DEFLT5 with your site ID in the Task Libraries fields.

- **Initial Commands**

Specifies a member in your INT1CLIB DD that includes commands to execute at the initial intercept.

Monitor Control Panel

Use the Monitor Control panel to specify which programs the application monitors during a debugging session. Explicitly define up to 30 programs, but the use of wildcards allows you to define an unlimited number of programs. The data that you enter on this panel remains in place from session to session until you delete or change the information.

Monitored Programs

Use the Monitored Programs section to define the programs that you want to debug. The program name must match the name of the PROTSYM member that contains the symbolic information for the program being debugged. You can define the complete name of up to 30 programs. You can specify more than 30 programs using when you use wildcards in the program name.

If you entered a **Y** in the Monitor PL/I field on the Monitor Control panel, the PL/I Protsym/Load Module Map panel appears when you press Enter. Specify the name of the load module in which these programs reside in the Modname field to the right of the entry.

Symbolic Files

Use the symbolic (PROTSYM) files section to define the files that contain the symbolic information for the programs that you want to debug. A PROTSYM entry must exist for each program that you want to debug. Specify at least one PROTSYM file in this section.

Dynamic Symbolic Support

Dynamic symbolic support dynamically retrieves the compiler or assembler listing of the monitored program from a listing data set managed by CA Endeavor SCM and loads it into the designated PROTSYM file. For this feature to work, the load module library (specified under Task Libraries on the Execution Control panel) where the load module is loaded and the listing data set containing the module listing must be under CA Endeavor SCM control.

The dynamic symbolic support feature always loads the correct symbolic information whenever a matching symbolic version is not found. To activate the dynamic symbolic support feature, enter **Y** to the right of the PROTSYM file, underneath the heading Endeavor Auto Populate to designate the PROTSYM file as the file to receive the listing. To deactivate the feature, change the **Y** to **N**.

Symbolic files associated with programs that do not contain a time stamp in the executable, such as non-LE-enabled Assembler programs, are reloaded every time they are monitored. If you want to suppress this behavior, specify **N** in the Always Auto-Populate Non-LE-Enabled Assembler field.



Note: The dynamic symbolic support feature cannot differentiate between multiple listing outputs created by a single CA Endeavor SCM processor for the same element. Dynamic symbolic support using listing outputs from multiple compiles or assemblies from a single CA Endeavor SCM GENERATE or MOVE action for the same element can produce unpredictable results.

Dynamic Symbolic Support Return Codes

Dynamic symbolic support provides an integrated service using API calls that deploy various proven components including CA Endeavor SCM, PROTSYM post processors, z/OS dynamic allocation, CSVQUERY, and binder services. The following table provides a list of possible return codes (RC) and reason codes (RSC) the API and various components can return.



Note: Other numeric return codes and reason codes are also returned by CA Endeavor SCM and z/OS binder services API (IEWBIND).

Function	RC	RSC	Meaning
API	0	0	Operations completed successfully.
API	2	0	GETMAIN failed.
API	8	0	CA Endeavor SCM server not activated.
API	12	0	Invalid or missing parameter in NDVRCOMM.
API	16	0	Bad return code from CA Endeavor SCM. See CA Endeavor SCM documentation.
API	20	0	Data set open failed.
API	28	0	Allocation error. See IBAPILOG file.

Function	RC	RSC	Meaning
API	32	0	IN25DALC load failed.
API	36	0	ENA\$NDVR load failed.
API	40	0	CA Endeavor SCM footprint not found for requested element.
API	44	0	Requested CSECT not found.
API	30	0	IN25CDRV load failed.
BINDER	44	0	Requested CSECT not found by binder.
CSVQ	48	0	CSECT from IDRU does not match CSECT in NDVR_NAM2.
Endevor	12	0	Incorrect C1DEFLT5 used.
Endevor	32	15	CA Endeavor SCM encountered critical error and cannot continue.
NDSB	4	0	Compressed footprint in IDRU invalid.
NIDR	2	0	GETMAIN failed.
NIDR	32	0	IN25NDSB load failed.
NSRV	8	NSRV	Listing server ABORTed or failed.
0000	12	0000	Incorrect C1DEFLT5 used.

Intercept Panel

The Intercept panel is the point where execution of the test program is controlled.

The Intercept panel has the following attributes:

- The monitor name of the program intercepted is displayed in the upper left corner of the panel.
- The reason the program has been intercepted is displayed in the title line of the panel (*INITIAL* INTERCEPT).
- The trace line shows the last ten statement numbers executed.
- The program listing displayed is scrolled to the current statement and that statement is highlighted.



Note: Each time a test begins, the application displays an Initial Intercept panel that stops before the first monitored executable statement is executed. At this point, use breakpoints and control commands before the program is executed.

You can specify the number of times each statement has been executed as shown in the following panel. This is called the *frequency counter*. For more information about the frequency counts, see [Control Commands \(see page 67\)](#).

Invoke all other test panels from the Intercept panel. You must return to the Intercept panel before accessing another test panel. For example, to format a data item, enter the Display command in the Command field. To return to the Intercept panel, press the END key. To resume execution, enter **GO** in the Command field, use the G line command, or press the appropriate PF key. The test continues until the next intercept condition is encountered, an ABEND occurs or the test comes to the normal end of job.

Types of Intercepts

The type of intercept being displayed is shown at the top center of the panel on the title line. The following list describes the possible intercepts.

- ***INITIAL* INTERCEPT**

This intercept is displayed upon entry into the first program that you have asked to monitor. To have the application stop upon entry to all programs, enter AT ALL ENTRY on the command line at this intercept.

- **UNCOND BEFORE and UNCOND AFTER INTERCEPT**

This intercept is displayed when the program reaches an executable statement where the application was instructed by the programmer to stop executing the program.

- **ABEND INTERCEPT**

This intercept is displayed when a monitored program ABENDs or an unhandled condition has been raised under LE/370. The ABEND code is shown in the title line. Use application commands to resolve ABENDs. For example, if invalid data is in a data item, use the set command to change the value. Continue the test using the GO command.

- **ATTENTION INTERCEPT**

This intercept is displayed when a potential breakpoint is reached after the ATTENTION or PA1 key is pressed. For more information on use of the ATTENTION or PA1 key, see the Stopping a Looping Program in Testing Procedures. Enter the GO command in the Command field or press the appropriate PF key.

- **STEP BEFORE and STEP AFTER INTERCEPT**

This intercept is displayed when the count parameter in the STEP command is matched. Enter the STEP command in the Command field of the Intercept panel. For more information about this command format, see STEP Control Command in Debugging Commands.

The STEP command specifies the number of statements to be executed each time the GO command is issued. The STEP COUNT Intercept panel is displayed after each execution.

- **NEXT BEFORE and NEXT AFTER INTERCEPT**

This intercept is displayed when the next count (set by the NEXT command) is exceeded. Enter the NEXT command in the Command field of the Intercept panel. For more information about this command format, see the NEXT Control Command in Debugging Commands. The NEXT command causes the execution of the next n statements in the program. The NEXT command operand specifies the number of statements to be executed before the Next Count Intercept panel is displayed again.

- **WHEN *when-name* BEFORE; WHEN *when-name* AFTER INTERCEPT**

This intercept is displayed at a conditional breakpoint. A conditional breakpoint is detected when the conditional statement specified in the COND command is true. Enter the COND command in the Command field of the Intercept panel to set conditional breakpoints or to access the When panel. [For more information about the command formats, see Control Commands \(see page 67\)](#). Set conditional breakpoints with the line commands C, W, and V.

- **LABEL INTERCEPT**

This intercept is displayed with executing a label when the AT ALL LABEL or AT LABEL breakpoint has been set.

- **DBCALL INTERCEPT**

This intercept is displayed when a call is made to a database, such as DB2 or IMS, when the AT ALL DBCALL or AT DBCALL breakpoint has been set.

- **PGM ENTRY INTERCEPT**

This intercept is displayed upon entry to any program that you are monitoring when the AT ALL ENTRY or AT ENTRY breakpoint has been set.

- **PGM EXIT INTERCEPT**

This intercept is displayed before exiting any COBOL program that is being monitored when the AT ALL EXIT or AT EXIT breakpoint has been set.

Breakpoint Panels

This article describes the breakpoint panels that you can access from Option 1 on the main menu.

- [Breakpoint Panel \(see page 27\)](#)
- [Breakpoint Status Panel \(see page 27\)](#)
- [When \(or COND\) Panel \(see page 28\)](#)
- [When Conditions Display Panel \(see page 30\)](#)
- [Equate Panel \(see page 30\)](#)
- [Equate Display Panel \(see page 31\)](#)

When you use a conditional or unconditional breakpoint command but do not specify any operands, the application displays a series of panels to collect the information needed to define the breakpoint.



Note: You can also set breakpoints by entering a single-character line command on the line where you want the breakpoint. There is no limit to the number of breakpoints you can set.



More information:

[Control Commands \(see page 67\)](#)

Line Commands (see page 46) .

Breakpoint Panel

Use the Breakpoint panel to set, reset (delete), or list unconditional breakpoints. Access the Breakpoint panel by entering the command UNCOND (or AT) or OFF, without operands, in the Command field of the Intercept panel.

Use the OPTION field to enter S (set), R (reset), or L (list):

- **S**

You must enter the statement number. The program stops whenever the statement number is encountered, unless you enter a count in the COUNT field.

You can use the COUNT field to specify the number of times a statement is executed before the program is stopped. The program stops at the designated statement prior to the final increment of the COUNT field. The commands that you specified to be executed at breakpoint are executed when the COUNT limit is satisfied.

Use the AFTER field to indicate whether you would like execution to stop before this statement is executed (N) or after the statement is executed (Y).

- **R**

Deletes a breakpoint. You must enter the statement number. To delete a breakpoint for another program, enter the program name.

- **L**

Displays the Breakpoint Status panel. This is a listing of all unconditional breakpoints currently set in the program.

You can specify application commands to execute at a breakpoint by entering the commands (separated by semicolons) in the COMMANDS TO BE EXECUTED AT BREAKPOINT field. The commands execute until the program resumes through the GO command, or until there is a request for a panel through a command such as DI or AT. The remaining commands are ignored. The list of commands can be up to 73 characters long and should be entered as shown in the following example:

```
COMMANDS TO BE EXECUTED AT BREAKPOINT:
====>SET BINARY-1 = 01;LDI BINARY-1;NEXT 5
```

In this example, the following commands are executed when the breakpoint occurs:

- The value of data item BINARY-1 is set to 1.
- Data item BINARY-1 is displayed on the Intercept panel in a *keep window*.
- Five statements are executed.

Breakpoint Status Panel

The Breakpoint Status panel lists breakpoints set in the Breakpoint panel and by the UNCOND or AT commands for the current program.

The panel displays the following information:

- **PROGRAM**
This is the monitored program name.
- **STMT#**
This is the statement number of the breakpoint.
- **COUNT**
This shows the number of times a breakpoint is executed before the program stops and the command in the COMMANDS TO BE EXECUTED field is executed.
- **EXEC**
This shows the number of times the breakpoint has been executed.
- **B/A**
This indicates whether execution should stop before or after the statement.
- **COMMANDS TO BE EXECUTED**
These are application commands specified in the Breakpoint panel or by the AT command.
- **SOURCE LINE**
This shows the source line where the breakpoint was set.



Note: Use an O or X line command to remove these breakpoints from this screen.

You can use all SCROLL keys and SCROLL variables in this display.

When (or COND) Panel

Use the When panel to set, reset (delete), or list conditional breakpoints. Access this panel by entering either COND (or WHEN) or OFFWN, without operands, in the Command field of the Intercept panel.

You give each When condition a *when-name*. This name is up to eight characters long. If the when-name specified is a statement number in the program, the when condition is checked only before executing this statement. This line-specific when condition is known as a local conditional and requires less computing overhead since the condition is not tested before every statement.

A When-condition breakpoint is defined by one of two conditions:

- If only the when-name and the data-area-1 operands are specified, a W when-name Intercept panel is displayed each time the value of the data item specified in data-area-1 changes. This type of conditional breakpoint is called a variable change breakpoint.
- If the operands when-name, data-area-1, operator, and data-area-2 are specified, the data item specified in data-area-1 is compared to the data item specified in data-area-2 based on the operator. When the condition is true, a W when-name Intercept panel is displayed. (Data-area-2 can be a data item, a constant, LOW-VALUES, HIGH-VALUES, SPACES, or ZEROES.)

Use the OPTION field to enter S (set), R (reset), or L (list):

- **S**
The when-name and data-area-1 operands are required; the operator and data-area-2 operands are optional.
- **R**
Only the when-name operand is required.
- **L**
Displays the When Conditions Display panel. This is a listing of all conditional breakpoints in the program.

Field Descriptions

- **WHEN NAME**
A 1-8 character name assigned to the test or program statement number.
- **DATA-AREA-1**
The data-name to be tested.
- **OPERATOR**
Condition which DATA-AREA-1 and DATA-AREA-2 must meet for the breakpoint to occur. Valid values are:
 - GT or > Greater Than
 - LT or < Less Than
 - EQ or = Equal
 - NE or ? = Not Equal
 - GE or >= Greater Than or Equal
 - LE or <= Less Than or Equal



Note: => and =< are not permitted.

- **DATA-AREA-2**
(Optional) Data-name or content which is compared with Data-AREA-1.
- **BEFORE/AFTER**
Field to indicate whether to stop before or after the statement when the condition is recognized.

Execute commands at a conditional breakpoint by entering the commands (separated by semicolons) in the COMMANDS TO BE EXECUTED AT WHEN CONDITION field. At the breakpoint, commands execute sequentially until the program resumes execution through the GO command, or until there is a request for a panel through a command such as the DI or UNCOND command. The remaining commands are ignored. The list of commands can be up to 73 characters long and should be entered as shown in the following example:

```
COMMANDS TO BE EXECUTED AT WHEN CONDITION:
====>SET BINARY-1 = 01;LDI BINARY-1;NEXT 5
```

When Conditions Display Panel

The When Conditions Display panel lists conditional breakpoints set in the When panel, by the V, C and W line commands, and by the COND (or WHEN) command for the current program with operands in the Command field of an Intercept panel.

A sample When Conditions Display panel follows:

```
----- CA InterTest Batch BREAKPOINTS PANEL -----
COMMAND ==>                                     SCROLL ==> CUR

***** Top of Data *****
CONDITIONAL BREAKPOINTS FOLLOW:
PROGRAM   WHN NAME  DATA NAME 1                OP   DATA NAME 2
-----
. CAMRCOBB LOOPCOND LOOP-OUT                      GT   +5000.
..SUBCOMMANDS FOR ABOVE WHEN ==> SET LOOP-OUT=0

***** Bottom of Data *****
```



Note: Use an O or X line command to remove the breakpoint from this screen.

You can use all SCROLL keys and SCROLL variables in this display.

Equate Panel

Use the Equate panel to set, drop (delete), or list equated variable names. You can equate a data item with a symbolic name and then refer to by either the symbolic name or the data name. There is no limit to the number of equates that can be set. Access the Equate panel by entering the command EQU or DROP, without operands, in the Command field of the Intercept panel.

Use the OPTION field to enter S (set), D (drop), or L (list):

- **S**

You must enter an equate name and the data-name-1 fields. You only need to enter the data-name-2 and data-name-3 fields when equating qualified data-items. Data-name-1 can be a subscripted or indexed data item.

- **D**

Drops an equate. You must enter the equate name.

- **L**

Displays the Equate Display panel. This is a listing of all equates that have been set using the EQUATE command



Note: For more information on the use and function of this command, see the EQUATE Control Command in [Debugging Commands](#) (see page 40) .

Equate Display Panel

The Equate Display panel lists the equate name and actual name for all equates currently set.

You can use all SCROLL keys and SCROLL variables in this display.

Data Display Panels

You view or alter data values in a monitored program using data display panels. Enter one of the following commands in the Command field of the Intercept panel to access each type of display.

- **DISPLAY**
Display a data item defined to the monitored program or the WORKING STORAGE SECTION of a COBOL program.
- **LINKAGE**
Display the LINKAGE SECTION of a COBOL program. This command is not valid for assembler or PL/I.
- **RECORDS**
Display the FILE SECTION file descriptions of a COBOL program. This command is not valid for assembler or PL/I.



Note: When the DATAMON command is in effect and the trace navigation commands (PREV, ADV, POINT) have been used since the last breakpoint, this panel does not allow altering of data values. In addition, the data values that are displayed correspond to the data values before the current trace entry was executed.

Display Panel

The Display panel shows the following information:

- **ADDRESS**
This column shows the starting virtual storage address for the data item. Use this address to address the data in Core, Option 2 on the PRIMARY OPTION MENU.
- **TYPE**
This is defined as follows:
 - Alphabetic
 - AN Alphanumeric
 - AN-V Varying alphanumeric
 - ANE Alphanumeric edited
 - BF Binary floating point

- BIT Bit
- BIT-V Varying bit
- Display (Sterling nonreport)
- DE Display edited (Sterling report)
- DBC Graphic or DBCS
- DBC-V Varying graphic
- Floating point (COMP-1/COMP-2)
- FD Floating point display (external floating point)
- HX Hexadecimal
- N National (UTF-16)
- NB Numeric binary unsigned (COMP)
- NB-S Numeric binary signed
- ND Numeric decimal unsigned (external decimal)
- ND-OL Numeric display, overpunch sign leading
- ND-OT Numeric display, overpunch sign trailing
- ND-SL Numeric display, separate sign leading
- ND-ST Numeric display, separate sign trailing
- NE Numeric edited
- NP Numeric packed decimal unsigned (COMP-3)
- NP-S Numeric packed decimal signed
- PTR Pointer data item (usage pointer)
- UI Index data item (usage index)
- Subscripted
- **LINE #**
This column shows the program line numbers where the data names are defined.
- **LV**
This column shows the level number generated by the compiler. Level numbers appear in the DMAP output of the COBOL compiler or the AGGREGATE output of the PL/I compiler.

NAME/VALUE

This column displays the data-item names.

The value of the data item is displayed as it is defined in the monitored program. If the data in the VALUE field is more than 20 characters long, it is shown on multiple lines.



Note: COBOL programs can be compiled with the ARITH(EXTEND) parameter, which will allow a numeric field to be greater than 18 bytes. These fields will be displayed with the message 'TOO BIG TO DISPLAY' in the value. To view or change the actual value of the field, use the CORE command.

Alter data by typing over the value field. The application truncates data if you attempt to modify a data item with a value larger than the size of the data item. Movement of new values to the data area is done according to how the item is defined. For example, if the variable is alphanumeric and is 2 bytes long, if you enter 12B, the value truncates to 12.

You can use all SCROLL keys and SCROLL variables.

For assembler programs, when an access register or 64-bit address is requested for display, an additional line shows the access register or “high half” value, as shown in the following panels:

```

----- CA InterTest Batch ASSEMBLER DATA DISPLAY -----
COMMAND ==>                                SCROLL ==> CUR
ADDRESS  TYPE  LINE # LV  NAME  / VALUE      0----,----1----,----2--
***** Top of Data *****
High-Half ==> 00000001
00000000 HX      000000 02 R1      0000000000000000
00000008 HX      000000      +0008  0000000000000000
00000010 HX      000000      +0016  00000003000B3958
*****
***** Bottom of Data *****
----- CA InterTest Batch ASSEMBLER DATA DISPLAY -----
COMMAND ==>                                SCROLL ==> CUR
ADDRESS  TYPE  LINE # LV  NAME  / VALUE      0----,----1----,----2--
***** Top of Data *****
ALET      ==> 00010004
00000000 HX      000000 02 R3?      0000000000000000
00000008 HX      000000      +0008  0000000000000000
00000010 HX      000000      +0016  0000000000000000
*****
***** Bottom of Data *****

```

For more information on Assembler indirect addressing, see [Control Commands \(see page 67\)](#).

Invoke the panel by entering the Display command, with or without operands, from an Intercept panel. For assembler and PL/I programs, operands are required.

Linkage Panel

To invoke the Linkage panel, enter the Linkage command with or without operands from an Intercept panel. This command is only valid for COBOL programs.

Display the Linkage panel after the first PROCEDURE DIVISION statement has been executed, otherwise the linkage addresses will not be resolved. This occurs only when the Intercept panel is displayed for the first time and the LINKAGE command is entered. You can avoid this by using the NEXT command the first time the Intercept panel is displayed to execute the first statement in the PROCEDURE DIVISION and resolve the linkage addresses.

Records Panel

To display the Records panel, enter the REcords command with or without operands from an Intercept panel. This command is only valid for COBOL programs. You can only use the RECORDS command, which displays the File Section Display panel, to modify data if the file is open.

File Status Panel

The File Status panel displays the characteristics of all assigned files, and shows whether they are open or closed. Enter the FILES (FI) command from an Intercept panel to open this screen. The File Status panel is only available for COBOL programs.

The following screen is an example of the File Status panel:

```
----- CA InterTest Batch FILE STATUS PANEL -----
COMMAND ===>                                SCROLL ===> PAGE
FD-NAME-----DDNAME--DSNAME-----
INPUT-FILE          SYSUT1  INTC.SOME.DATAS
OPEN  INPUT  DSORG=PS RECFM=FB      BLKSIZE=06000 LRECL=0080
DCB=00123456  DEB=006C941C
SYMBOL-FILE          SYMBOL  *** WARNING *** DD CARD NOT ALLOCATED
CLOSED          DSORG=PS RECFM=..... BLKSIZE=00800 LRECL=0080
DCB=0018D42C
OUTPUT-FILE          SYSUT2  INTC.NOTA.PDS
OPEN  OUTPUT  DSORG=PS RECFM=FB      BLKSIZE=06000 LRECL=0080
DCB=0018DD40  DEB=006C941C  DECB=0018BD80
. . . . .
```

If you receive a warning that says the DD CARD is not allocated take one of the following actions:

- Split the screen and enter TSO using the ALLOCATE command to allocate the required file.
- Enter the TSO ALLOCATE command on the command line of the current ISPF screen.
- Invoke another session and select Option 3 to allocate the file.
- Suspend the session and perform the allocation under CA Roscoe, for CA Roscoe users. Then resume the ETSO session to resume your CA InterTest Batch session.



Note: Batch Link users must use the DDALLOC command to allocate the DD. For more information on this command, see [Control Commands \(see page 67\)](#).

The data event block (DEB) is only displayed for open BSAM files.

For OS/VS COBOL users, IMS databases do not appear on the status display.

Program Trace Display Panel

The Program Trace Display panel displays program statements in the order of their execution. To initiate a trace, enter the TRACE or TRACE SOURCE command in the Command field of the Intercept panel. For more details on the command format for TRACE Control Command, see [Debugging Commands](#) (see page 40).

A sample Program Trace Display panel with TRACE follows:

```
----- CA InterTest Batch PROGRAM TRACE DISPLAY -----
COMMAND INPUT ==>                                SCROLL==> PAGE
StmtNo  Program  Label Name/Source Line
043700   CAMRCOBB
043500   CAMRCOBB
043400   CAMRCOBB  FORCING-THE-ABEND
043300   CAMRCOBB
043100   CAMRCOBB
042200   CAMRCOBB
042034   CAMRCOBB
042033   CAMRCOBB
042031   CAMRCOBB
051500   CAMRCOBB  0100-SOME-MEANINGFUL-LABEL
051400   CAMRCOBB
050600   CAMRCOBB
050400   CAMRCOBB
050200   CAMRCOBB
049800   CAMRCOBB  0200-THESE-ARE-PARAGRAPH-NAMES
049700   CAMRCOBB
049400   CAMRCOBB
048600   CAMRCOBB
042030   CAMRCOBB
042029   CAMRCOBB
042028   CAMRCOBB
```

A sample Program Trace Display panel with TRACE SOURCE follows:

```
----- CA InterTest Batch PROGRAM TRACE DISPLAY -----
COMMAND ==>                                Scroll ==> PAGE
StmtNo  Program  Label Name/Source Line
***** TOP OF DATA *****
001042   CAMRCOBB      MOVE R1498-TIME0 TO R1498-TIME.
001041   CAMRCOBB      SPLIT-SCREEN.
001038   CAMRCOBB      IF S-AID = 'PF3 ' OR 'PF15'
001037   CAMRCOBB      CALL 'ROETSAPI' USING GETV S-AID-DEF S-AID.
001035   CAMRCOBB      IF RETURN-CODE NOT EQUAL ZERO
001034   CAMRCOBB      CALL 'ROETSAPI' USING PANL PANL-PARM.
001033   CAMRCOBB      MOVE '73' to PANL-ID.
001028   CAMRCOBB      DEMONSTRATE-SPLIT-SCREEN.
000950   CAMRCOBB      GO TO DEMONSTRATE-SPLIT-SCREEN.
000949   CAMRCOBB      IF P-OPT = '3'
000947   CAMRCOBB      IF P-OPT = '2'
000945   CAMRCOBB      IF P-OPT = '1'
000942   CAMRCOBB      IF S-AID = 'PF3 ' OR 'PF15'
000939   CAMRCOBB      IF RETURN-CODE NOT EQUAL ZERO
000937   CAMRCOBB      CALL 'ROETSAPI' USING GETV S-AID-DEF S-AID
000934   CAMRCOBB      IF RETURN-CODE NOT EQUAL ZERO
000933   CAMRCOBB      CALL 'ROETSAPI' USING PANL PANL-PARM.
000932   CAMRCOBB      MOVE '07' to PANL-ID.
000930   CAMRCOBB      MOVE ZEROES TO LOOP-OUT.
000929   CAMRCOBB      OPTIONS.
```

The Command field accepts only display commands. For valid commands, see [Display Commands \(see page 40\)](#). You can use all SCROLL keys and SCROLL variables in this panel.

Core Option

Use the Core option, option 2 on the Primary Option Menu, to display the memory for your TSO user region during a test. You can also use the CORE command from any Intercept panel.

The screen displays memory in hexadecimal and character formats. You can change memory that is not modify-protected by overwriting the data on the display. Fetch protected memory cannot be displayed or modified.

Locate data in memory by using the following methods:

- The FIND commands search for a specified character or hexadecimal string.
- The DISPLAY command locates a specified address.
- Specify a hexadecimal and use the scroll keys to shift the contents of the display by the specified amount.
- Type an address over the primary address field, or an offset from the primary address can be typed over the primary offset field. The primary offset fields are and the display scrolled to that part of memory.

You can also use the following PF keys:

- **PF10**
Restore the address that was previously displayed.
- **PF11**
The address indicated by the cursor position appears in the primary address field and the display is scrollable to that part of memory.



Note: If you enter the CORE command from an Intercept screen, the display includes an updateable EXPR field, which shows a register expression or the address currently displayed. Commands that modify the displayed address also modify this field.

DISPLAY

The DISPLAY command sets the primary address field to the specified address. The primary address field is updated to the indicated address in virtual storage. The primary offset field is set to zero.

Syntax

DISPLAY *address*

- **address**
Specify the address in virtual storage. The address is case-sensitive and can be a one to eight digit hexadecimal number.

FIND Command

The FIND command, or F, searches memory until the string operand is matched.

Syntax

FIND *string* <NEXT|FIRST|LAST|PREV>

- **string**
Specify the search argument on the data area. The string can be alphabetic, numeric, and special characters. Use single or double quotation marks around the string if it contains embedded blanks or quotes.
- **NEXT**
Search for the next occurrence of the string, starting at the beginning of the first address displayed.
- **FIRST**
Scroll to location zero followed by a FIND NEXT operation.
- **LAST**
Start the scan at address 16,777,215 (X'FFFFFF') and move backwards to find the last occurrence of string.
- **PREV**
Start the scan at the current address and move backward to find the previous occurrence of the string.

FX (Find Hex Data)

The FX command searches memory until the hexadecimal string is matched.

Syntax

FX *string* <NEXT|FIRST|LAST|PREV>

- **string**
Specify a search argument of one to eight hexadecimal characters.
- **NEXT**
Search for the next occurrence of string starting at the beginning of the first address displayed.
- **FIRST**
Scroll to location zero, and then process the FIND NEXT operation.

- **LAST**
Start the scan at address X'FFFFFF' (16,777,215) and move backwards to find the last occurrence of string.
- **PREV**
Start the scan at the current address and move backwards to find the previous occurrence of string

POP

The POP command restores the address most recently pushed onto the stack. The address becomes the primary address on the display, and the primary offset field is set to zero. If the stack is empty, an EMPTY STACK message appears.

Syntax

POP

PUSH

The PUSH command places on the stack the current primary address.

Syntax

PUSH

Map Option

The Region Map Display panel shows the task control blocks (TCBs) for the program that you initiate the command from. Launch this panel by using the MAP command from any debugging session. You can also use Option 4 from the primary option menu.

The Region Map Display panel shows the following information about the TCB list:

- **TCB**
The address of the current TCB
- **ID**
The number associated with the current TCB



Note: Use the ID command to overwrite this field.

- **TIOT**
The task input/output table

- **ASCB**
The location of the address space control block
- **ASID**
The address space identifier

The table shows the following information for each TCB listed:

- **TCB**
The name of each TCB that resides under the parent
- **ID**
The ID number for each TCB listed
- **Level**
The hierarchical representation of the TCB
- **Program**
The program the TCB is assigned to
- **Status**
The state of the TCB
- **NDISP**
The format

Use the following commands to control what TCB information the Region Map Display panel shows.

- **ID *task number***
Set the TCB number for your program.
- **CDE**
The programs in storage for the current TCB ID number.
- **SAVE**
The save area map for the current TCB ID number.
- **ALLOC**
The current allocations by ddname. The option ALLOC is similar to the TSO LISTALC command, but ALLOC displays the ddname. You can also scroll the display.
- **FILE**
The Information on open files related to the current TCB ID number.



Note: IMS database files do not display using the FILE command. Use the ALLOC command to see information for IMS databases.

- **TCB**
The status of the user's TCBs (not task related).

- **RB**
The active request blocks (RB) for the current TCB ID number.
- **ALLTCB**
All TCBs for your TSO session

Use the scroll keys to view and select the TCB number of the program you are testing from the list of ID numbers on the Control Block Display panel.

Debugging Commands

This section describes the commands valid for use during a debugging session. There are four types of commands:

- Display
- Line
- Control
- Report

Use *display* commands to locate selected data in the program and position it on the screen. These are commands such as FIND and SCROLL; they can be used in any panel containing a command input field.

Control and *line* commands direct the test session and you enter them only in the Intercept panel. Line commands are shorter versions of some control commands.

Report commands generate reports to be printed and analyzed offline. Only enter report commands in the Intercept panel.

Display Commands

This article describes the display commands that you can use to locate and position data on the screen. Enter them on the command line of any panel containing a command input field.

- [COLUMN Command \(see page 41\)](#)
- [FIND Command \(see page 41\)](#)
- [FP Command \(Find Paragraph\) \(see page 42\)](#)
- [FS Command \(Find Statement\) \(see page 42\)](#)
- [KDOWN Command \(Scroll Keep Window Down\) \(see page 43\)](#)
- [KSIZE \(or KS\) Command \(Keep Window Size\) \(see page 44\)](#)
- [KUP Command \(Scroll Keep Window Up\) \(see page 44\)](#)
- [LOCATE Command \(see page 45\)](#)
- [.label Command \(see page 46\)](#)

COLUMN Command

The COLUMN command places a line that shows the column numbers on the first line of the display area of the Intercept or Display panel.

Syntax

Use the following syntax for COLUMN:

COLUMN <ON|OFF>

Variables

Use the following variables with the COLUMN command:

- **ON**
Enables the column numbers display.
- **OFF**
Disables the column numbers display.

COLUMN OFF is functionally equivalent to the RESET command, which removes the column numbers from the panel display.

FIND Command

The FIND command searches the file until the string operand is matched.

Syntax

Use the following syntax for FIND:

FIND string <col1 <col2>> <NEXT|FIRST|LAST|PREV>

You can enter FIND in its abbreviated form: F.

Variables

Use the following variables with the FIND command:

- **string**
The search argument on the data area. String can be alphabetic, numeric, and special characters. If string contains embedded blanks or quotes, it must be enclosed in single or double quotation marks.
- **col1**
Specifies the column in which to begin the search for string. If col1 is specified and col2 is omitted, the string must start in the specified column.
- **col2**
Specifies the last column of a range of columns to search for string. If col2 is greater than the record length, the record length is used.

- **NEXT**
Causes a search for the next occurrence of string starting at the beginning of the first line displayed after the cursor location. The default is NEXT.
- **FIRST**
Causes a scroll to the top of the data followed by a FIND NEXT operation.
- **LAST**
Starts the scan at the bottom of the data and moves backward to find the last occurrence of string.
- **PREV**
Starts the scan at the current cursor location and moves backward to find the previous occurrence of string.

Example:

To find the next occurrence of *TEST* in a program, enter:

```
F ''TEST''
```

FP Command (Find Paragraph)

The Find Paragraph command scrolls the display to the definition of the specified paragraph, procedure, or label name.

Syntax

Use the following syntax for FP:

```
FP paragraph-name|procedure-name|label-name
```

You must enter this command in its abbreviated form: FP.

Variables

Use the following variables with the FP command:

- **paragraph-name**
Designates a paragraph name defined in the Procedure Division of the currently qualified COBOL program
- **procedure-name**
Designates a procedure name defined in the currently qualified PL/I program.
- **label-name**
Designates a label name defined in the currently qualified assembler or PL/I program.

FS Command (Find Statement)

The Find Statement command scrolls a display to a specified statement number.

Syntax

Use the following syntax for FS:

FS statement-number

You must enter this command in its abbreviated form: FS.

Variable

- **statement-number**
Specifies a statement number in the currently qualified program.

Cursor Sensitive Find

If a data name is displayed and you wish to locate its definition, place the cursor over the data name and press Enter. The application repositions the display at the definition of the data name.

If a paragraph, procedure, or label name is displayed and you wish to locate the line on which it starts, place the cursor over the name and press Enter. the application repositions the display at the paragraph, procedure, or label.

KDOWN Command (Scroll Keep Window Down)

Use the KDOWN command to scroll the keep window in the downward direction. The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for KDOWN:

KDOWN scroll-amt

Variables

Use the following variables with the KDOWN command:

- **scroll-amt**
Identifies the number of lines to scroll, specified as one of the following:
- **PAGE**
Scroll up one full page.
- **HALF**
Scroll up one half page.
- **MAX**
Scroll up to the top of the window.
- **nnn**
Scroll the number of lines specified.

Usage Notes

The keep window is not scrollable if the number of entries in the window is less than the maximum depth defined by the KSIZE command. If the window is not scrollable, the KDOWN command is not functional.

If no argument is specified, the default scroll amount is used. The default scroll amount is defined in the upper right corner of the Intercept panel.

KSIZE (or KS) Command (Keep Window Size)

Use the KSIZE command to determine the size of your keep window. The keep window appears on the Intercept panel immediately above the first line of the program listing.

The KSIZE command lets you control the maximum depth of the keep window and whether the window should be fixed or variable in size.

Syntax

Use the following syntax for KSIZE:

KSIZE depth F|V

Enter this command in its abbreviated form: KS.

Variables

Use the following variables with the KSIZE command:

- **depth**
The maximum number of lines of data to be displayed in the window at one time.
- **F**
Specify F for a fixed window size.
- **V**
Specify V for a variable sized window.

Usage Notes

You can set the keep window size to any depth from 1 to 15 lines of data. The default depth is 6 lines.

If KSIZE is defined as fixed, the keep window always occupies the maximum depth, even if the window contains less data items than the maximum. If the window is defined as variable, it will never occupy more lines on the screen than are necessary to display all of the data items, and never more than the maximum depth.

Define your keep window as fixed to prevent bouncing of your source listing while stepping through the program.

KUP Command (Scroll Keep Window Up)

Use the KUP command to scroll the keep window in the upward direction. The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for KUP:

```
KUP scroll-amt
```

Variables

Use the following variables with the KUP command:

- **scroll-amt**
Identifies the number of lines to scroll, specified as one of the following:
- **PAGE**
Scroll up one full page.
- **HALF**
Scroll up one half page.
- **MAX**
Scroll up to the top of the window.
- **nnn**
Scroll the number of lines specified.

Usage Notes

The keep window is not scrollable if the number of entries in the window is less than the maximum depth defined by the KSIZE command. If the window is not scrollable, the KUP command is not functional.

If no argument is specified, the default scroll amount is used. The default scroll amount is defined in the upper right corner of the Intercept panel.

LOCATE Command

Use the LOCATE command to scroll to a specified line or a user-defined symbolic label in the program.

Syntax

Use the following syntax for LOCATE:

```
LOCATE nnnnnn|.label
```

You can enter this command in its abbreviated form: L.

Variables

Use the following variables with the LOCATE command:

- **nnnnnn**

Specifies a relative line number. The display scrolls so that the line number is the first line of the display. A line number of zero causes a scroll to the top of the data; 999999 causes a scroll to the bottom of the data.

- **.label**

Causes the display to scroll so that the line number equated to the symbolic label is the first line in the display.

A symbolic label equates to a line number. To establish a symbolic label, scroll the display until the line number to be equated is on the first line of the display. Enter the label in the COMMAND input field in the format:

xxxxxxx

The period is required and is followed by up to seven alphanumeric characters. The application does not retain symbolic labels when the test session ends.

.label Command

This command assigns a symbolic label to the current line number in the program.

Syntax

Use the following syntax for .label:

.xxxxxxx

Variables

Use the following variables with the .label command:

- **xxxxxxx**

The period is required and may be followed by up to 7 alphanumeric characters.

The application does not retain symbolic labels when the test session ends.

Line Commands

Line commands that you type in the far left field of a line to perform various actions. You can type multiple commands on different lines before pressing Enter, which processes all commands. Scrolling or any other command that causes the display to change also causes the specified line commands to be processed.

This article describes the following commands:

- [A or U Command \(Set Unconditional "Before" Breakpoints\) \(see page 49\)](#)
- [C or W Command \(Set Conditional Breakpoint\) \(see page 50\)](#)
- [D Command \(Display Working Storage\) \(see page 51\)](#)
- [G Command \(Go from Line\) \(see page 51\)](#)
- [H Command \(Display Hex Data\) \(see page 52\)](#)

- **I Command (Display Full Frequency Count Information)** (see page 53)
- **L or K Command (Display Data Item in a Keep Window)** (see page 54)
- **O or X Command (Turn Off Breakpoints)** (see page 55)
- **P Command (Print Display)** (see page 57)
- **R Command (Reset Data Item Display)** (see page 57)
- **S Command (Skip this Statement)** (see page 58)
- **V Command (Set Variable Change Breakpoint)** (see page 59)
- **) Command (Set Unconditional "After" Breakpoints)** (see page 60)
- **+ (plus) Command (Increment a Subscript)** (see page)
- **- (minus) Command (Decrement a Subscript)** (see page 64)
- **< nnnn (Scroll Left) Command** (see page 65)
- **> nnnn (Scroll Right) Command** (see page 66)

The following table is an overview of the Line commands, the corresponding Control Commands, and their function. The following pages provide a more complete description of each command.

Note that some commands have synonyms, such as **A** and **U**, which are fully interchangeable. Use whichever one you find easiest to remember.

Line	Control	Description
Commands	Commands	
A	AT	An unconditional "before" breakpoint is set at the specified line. This type of breakpoint stops <i>before</i> the specified program statement. You can enter an
U	UNCOND	"after" breakpoint by entering the) line command.
		A U remains in the first column to show where breakpoints are set.
		For assembler, if an A is placed next to a line in the keep window, the access register for that item is displayed.
C	COND	A conditional breakpoint is set at the specified line. Issuing this command causes the application to display the When screen, where the condition may be specified.
W	WHEN	The application automatically completes the line number portion of the When screen based on your breakpoint line.
D	DISPLAY	The DISPLAY panel is shown for all of the data items on the line where the D command is specified.
		For assembler, the operands for the current instruction in register notation are shown in the DISPLAY panel.
		Cursor-sensitive DISPLAY - If D is entered and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter displays the DISPLAY panel for only that item and data items to the <i>right</i> of the item on the same line.
G	GO	The program begins executing (Go to) the line where the G is typed.
H	KEEPX	The data items on the line where the H is placed are listed at the top of the Intercept panel in a keep window and shown in Hex format.
	LDIX	

Line	Control	Description
Commands	Commands	
	LDX	For assembler, the operands for the current instruction in register notation are added to the keep window. Cursor-sensitive KEEPX - if H is entered and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line. For assembler programs, if an H is placed on a line within the keep window, the high half of the 64bit register is displayed.
I	INFO	In instances where a frequency count exceeds 9999, Info displays the actual count at the top right of the screen. For assembler programs, if an I is placed on a line within the keep window, the access register is displayed.
K	KEEP	The data items on the line where the K or L is placed are listed in a keep window on the top of the Intercept panel.
	LDI	For assembler, the operands for the current instruction in register notation are added to the keep window. Cursor-sensitive KEEP - If you enter K and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line.
O , blank,	OFF	The breakpoint currently set on the line is reset (turned off).
X	OFFU	
P	PRINT	The DISPLAY for all the COBOL or PL/I data items on the line are written to the session log for later printing. For assembler, the operands for the current instruction in register notation are written to the session log for later printing.
R	REMOVE	An RDI, reset data item, is done for each of the data items on the line to remove entries in the keep window. You may also use it next to individual keep window entries.
	LDI	
S	SKIP	Skip this statement. The S remains in the first column to show you which statements are skipped.
V	COND	A variable change breakpoint is set at the specified line. The breakpoint is triggered before the line is executed when the application determines that the value of a variable in the line changes.
	WHEN	Cursor-sensitive variable change breakpoint - If you enter V and the cursor is placed on a COBOL or PL/I data item, pressing Enter designates that item as the variable change item.
X	OFF	Remove breakpoint or remove data item from keep window.
	RDI	
)	AT	

Line	Control	Description
Commands	Commands	
	UNCOND	An unconditional "after" breakpoint is set at the specified line. This type of breakpoint stops <i>after</i> the specified program statement. You can enter a "before" breakpoint by entering the U or A line command.
	A)	remains in the first column to show where breakpoints are set.
+ nnn		Increment a subscript by <i>nnn</i> on a displayed data item.
- nnn		Decrement a subscript by <i>nnn</i> on a displayed data item.
<[nnnn]		Used on a keep window entry scrolls the data left <i>nnnn</i> bytes.
>[nnnn]		Used on a keep window entry scrolls the data right <i>nnnn</i> bytes.

A or U Command (Set Unconditional "Before" Breakpoints)

The A or U line command performs the same function as the AT or UNCOND command. An unconditional "before" breakpoint is set on the line where the A or U is typed. The program stops at the statement prior to executing it. A U displays on the line to show you where breakpoints have been set. The U or A line command is only valid on executable lines of the program listing. Executable lines contain a frequency counter number or arrow. There is no limit other than the size of the screen. If you press Enter or the scroll keys, it sets the breakpoints.

You can type other commands over the U. Reset breakpoints using the OFF, O, X, or blank line commands.

The following panel shows "before" breakpoints being set on lines 13100, 13400, and 13736 of the program:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>                                     SCROLL==> PAGE
TRACE= 012400 012300
      7          CAMRCOB2          15.49.31          JUN 30,2014
0001    012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
--->    012400 DEMO-BEGIN.
--->    012500 GO TO DEMO-INIT.
--->    012600 DEMO-INIT.
--->    012700 OPEN OUTPUT OUTPUT01.
--->    012800 MOVE HD1 TO LINE-OUT.
--->    012900 PERFORM WRITE-LINE.

u--->    013100 FORCING-THE-ABEND.
--->    013200 MOVE MASK1 TO BIN1.
--->    013300 MOVE ZERO TO MASK1.
u--->    013400 MOVE MASK1 TO BIN1
          013410 BIN2
          013500 BIN3
          013600 BIN4.
--->    013700 TABLE-MOVE.
u--->    013736 MOVE TABLE-FILL TO WA-TABLE-1.
--->    013737 NUM-EDITED-TEST.
--->    013738 MOVE IN-FUNC TO OUT-FUNC.
--->    013739 MOVE IN-FUNC TO OUT-FUNC2.

```

Press Enter to set the breakpoints.

The listing appears as shown in the following panel. The lines that have "before" breakpoints set display a U in the first column to show you where the breakpoints are.

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept ----- COMMAND
====>                                     SCROLL====> PAGE
TRACE=> 012400 012300
          7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
--->      012400 DEMO-BEGIN.
--->      012500 GO TO DEMO-INIT.
--->      012600 DEMO-INIT.
--->      012700 OPEN OUTPUT OUTPUT01.
--->      012800 MOVE HD1 TO LINE-OUT.
--->      012900 PERFORM WRITE-LINE.

U--->      013100 FORCING-THE-ABEND.
--->      013200 MOVE MASK1 TO BIN1.

--->      013300 MOVE ZERO TO MASK1.
U--->      013400 MOVE MASK1 TO BIN1
          013410 BIN2
          013500 BIN3
          013600 BIN4.
--->      013700 TABLE-MOVE.
U--->      013736 MOVE TABLE-FILL TO WA-TABLE-1.
--->      013737 NUM-EDITED-TEST.
--->      013738 MOVE IN-FUNC TO OUT-FUNC.
--->      013739 MOVE IN-FUNC TO OUT-FUNC2.

```

You can also use the LISTAT command to show the Breakpoints panel as follows:

```

----- CA InterTest Batch BREAKPOINTS PANEL -----
COMMAND ==>                                     SCROLL ==> CUR

***** Top of Data *****
UNCONDITIONAL BREAKPOINTS FOLLOW:
PROGRAM  STMT#  COUNT  EXEC  B/A  COMMANDS TO BE EXECUTED
-----
. CAMRCOB2 000589 0000 0000  B
  SOURCE LINE: SPLIT-SCREEN.

. CAMRCOB2 000541 0000 0000  B
  SOURCE LINE: PAY-CALC.

. CAMRCOB2 000659 0000 0000  B
  SOURCE LINE: ORDER-CALC.
***** BOTTOM OF DATA *****

```

C or W Command (Set Conditional Breakpoint)

Use the C or W command to set a conditional breakpoint for a specified line. When you enter a C or W on a line and press Enter, the When screen displays. The WHEN-NAME field is populated with the line that you have specified. Use the When screen to specify the condition for the breakpoint. The condition is only checked before executing that statement. The application automatically completes the line number portion of the When screen based on your breakpoint line.

The C or W line command is only valid on executable lines in the program listing. Executable lines contain a frequency counter number or arrow.



Note: For further explanation of the When panel, see [Debugging Panels \(see page 19\)](#).

Example:

In the following example, you have entered the C command at line 1149 to set a conditional breakpoint.

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
Command ==> Scroll==> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
U0001 001141 ORDER-CALC.
0001 001142 MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
0001 001143 MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-0
0001 001144 PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
001145 UNTIL SUB-6 EQUAL 5.
U--> 001146 ORDER-CALC-EX.
--> 001147 GO TO OPTIONS.
0101 001148 TOTAL-ORDER.
c0101 001149 ADD 1 TO SUB-6.
0101 001150 ADD 1 TO LOOP-OUT.
0100 001151 MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
001152 GIVING ITEM-TOTAL.
0100 001153 ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
0100 001154 MOVE 1 TO SUB-6.
0100 001155 TOTAL-ORDER-EX.
001156 *-----*
001157 * The code above will loop. The loop can be *
001158 * interrupted by pressing an attention key.. .. *
001159 * The frequency counter should provide a clue *
```

When you press Enter in this example, the When panel displays, and the WHEN-NAME field is populated with the number 1149.

D Command (Display Working Storage)

For COBOL and PL/I programs the D line command causes the Display panel to be shown for all the data items on the line where the command is typed. For assembler programs, the D line command displays the operands for the current instruction in register notation. The D line command is valid on any line in the program listing.

You may type multiple D line commands on different lines, but to display the Display panel from each line in succession, press the END (PF3 or PF15) key. If D is specified on four lines, the Display panel appears for the data items on the first line. When END is pressed, the Display panel appears for the data items on the second line. This continues until the last D command is processed. Then the Intercept panel is displayed.

Cursor-sensitive Display: If you enter D and the cursor is placed on a COBOL and PL/I data item in the line, pressing Enter displays the Display panel for only that item and data items to the right of the item on the same line.

For more information on the Display Panel, see [Data Display Panels \(see page 31\)](#).

G Command (Go from Line)

The G line command causes the program to begin executing from the line on which you typed the G. Specify only one G line command at any time. If you type, execution begins from the highest line number. For example, if you type G on line 12900 and on line 13100, execution resumes at line 13100.

If you type G on line 13100 of the listing, as shown in the following panel, the program begins executing on line 13100 instead of normally beginning on line 12800. Lines 12800 and 12900 are not executed.

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>                                           SCROLL==> PAGE
TRACE=> 012800 012700 012600 012400 012300
          7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
--->      012500 GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
U--->     012800 MOVE HD1 TO LINE-OUT.
--->     012900 PERFORM WRITE-LINE.

g--->     013100 FORCING-THE-ABEND.
--->     013200 MOVE MASK1 TO BIN1.
--->     013300 MOVE ZERO TO MASK1.
U--->     013400 MOVE MASK1 TO BIN1
          013410 BIN2
          013500 BIN3
          013600 BIN4.
--->     013700 TABLE-MOVE.
U--->     013736 MOVE TABLE-FILL TO WA-TABLE-1.
--->     013737 NUM-EDITED-TEST.

```

H Command (Display Hex Data)

For COBOL or PL/I programs, the H line command causes a data item to be displayed in hexadecimal format for any data items contained on the line where you type the H. See the LDX command in the Control Commands section that follows for further explanation of the display. For assembler programs, the H line command displays the operands for the current instruction in register notation. For assembler programs, there is no functional difference between the H line command and the K line command. The number of H's that you specify is limited only by the number of data items displayed, because of the screen size.

Cursor-sensitive hex display: If you enter H and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter displays in hexadecimal format only that data item and data items to the right of that item on the same line.

You may reset the hex displays using the RDI command or the R line command.



Note: You cannot modify the zones and numerics portion of the data display. You also cannot modify group items.

In the following panel, the program has stopped executing because of an S0C7 abend as shown at the top of the Intercept panel. You have typed an H on line 13200 where the ABEND occurred to view the data items from that line. If you press Enter, it causes the line command to process.

```

CAMRCOB2 ----- CA InterTest Batch ABEND S0C7 Intercept -----
COMMAND ==>                                           SCROLL==> PAGE
TRACE=> 013200 013100 020100 020000 012900 012800 012700 012600 012500 012400
          7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500 GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
0001      012800 MOVE HD1 TO LINE-OUT.

```

```

0001      012900      PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
h0001      013200      MOVE MASK1 TO BIN1.
--->      013300      MOVE ZERO TO MASK1.
U--->      013400      MOVE MASK1 TO BIN1
          013410          BIN2
          013500          BIN3
          013600          BIN4.
--->      013700 TABLE-MOVE.
U--->      013736      MOVE TABLE-FILL TO WA-TABLE-1.
--->      013737 NUM-EDITED-TEST.

```

The following panel shows the hex data displays for the data items, MASK1 and BIN1, on line 13200. From the displayed data items, you should be able to determine the cause of the ABEND. A character field, MASK1, filled with blanks (X'40') is being moved to a numeric binary field. You could then use the SET command to change MASK1 to an acceptable numeric value.

```

CAMRCOB2 ----- CA InterTest Batch ABEND S0C7 Intercept -----
COMMAND ==>                                           SCROLL==> PAGE
TRACE=> 013200 013100 020100 020000 012900 012800 012700 012600 012500 012400
-----
00243DA0 AN      004000 03 MASK1
          DATA OCCUPIES 0002 BYTES OF STORAGE          ZONES 44
                                                    NUMERICS 00

00243DB2 NB-S    004600 03 BIN1
          DATA OCCUPIES 0004 BYTES OF STORAGE          ZONES 0000
                                                    NUMERICS 0000
-----
7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMS TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500      GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700      OPEN OUTPUT OUTPUT01.
0001      012800      MOVE HD1 TO LINE-OUT.
0001      012900      PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.

0001      013200      MOVE MASK1 TO BIN1.
--->      013300      MOVE ZERO TO MASK1.
U--->      013400      MOVE MASK1 TO BIN1

```

For assembler programs, if you use the H line command within the keep window, the high half of the 64bit register is displayed in a short message at the top of the screen. The following panel shows the short message displayed when the H command is used in the keep window:

```

MEMTEST ----- CA InterTest Batch *INITIAL* Inte- High Half = 00000001
COMMAND ==>                                           SCROLL ==> CSR
TRACE=> 000060 000059 000058 000056 000055 000050 000049 000048 000047 000046 0
-----
00000000 HX      000000 02 R1                      000000000000000000
-----
0001 000096 900F 1000          00000 59          STM R0,R15,0(R1)
---> 00009A EB0F 1040 0026          00040 60          STMH R0,R15,64(R1)
---> 0000A0 9B0F 1080          00080 61          STAM R0,R15,128(R1)

```

I Command (Display Full Frequency Count Information)

In instances where a frequency count for a line exceeds 9999, the I line command displays the actual count at the top right of the screen. After typing I to the left of the line, press Enter to display the frequency count.

The following panel shows the actual frequency count for a line number that has exceeded 9999 executions:

```

CARMC0B2 ----- CA InterTest Batch WHEN LOOPCOND BEFORE Inte Executed 10000
Command ==> Scroll==> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
U0001 001141 ORDER-CALC.
0001 001142 MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
0001 001143 MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-0
0001 001144 PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
001145 UNTIL SUB-6 EQUAL 5.
U---> 001146 ORDER-CALC-EX.
---> 001147 GO TO OPTIONS.
0101 001148 TOTAL-ORDER.
I010K 001149 ADD 1 TO SUB-6.
0101 001150 ADD 1 TO LOOP-OUT.
0100 001151 MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
001152 GIVING ITEM-TOTAL.
0100 001153 ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
0100 001154 MOVE 1 TO SUB-6.
PP 5668-958 IBM VS COBOL II Release 3.2 09/05/90 COB2DEMR

```

For assembler, if you use the I line command within the keep window the access register of the item is displayed in a short message at the top of the screen. The following panel shows the short message displayed when you use the A command in the keep window.

```

TESTAR ----- CA InterTest Batch ABEND S0C7 Int- ALET ==> 00010004
COMMAND ==> SCROLL ==> CSR
TRACE=> 000060 000059 000058 000056 000055 000050 000049 000048 000047 000046 0
-----
00000000 HX 000000 02 R1 00000000000000000000.
-----
0001 000096 900F 1000 00000 59 STM R0,R15,0(R1)
---> 00009A EB0F 1040 0026 00040 60 STMH R0,R15,64(R1)
---> 0000A0 9B0F 1080 00080 61 STAM R0,R15,128(R1)

```

L or K Command (Display Data Item in a Keep Window)

For COBOL or PL/I programs, the L line command causes a data item to be displayed on the Intercept panel for any data items contained on the line where the L is typed. For assembler programs, the L line command displays the operands for the current instruction in register notation. This is commonly known as a *keep window*. For more information on LDI commands, see the LDI (or KEEP) command in the Control Commands section that follows. The number of L's you can specify is limited only by the number of data items displayed because of the screen size.

Cursor-sensitive keep: If you enter L or K and the cursor is placed on a COBOL or PL/I data item in that line, pressing Enter lists in the keep window only that data item and data items to the right of that item on the same line.

The LDI (or keep window) displays may be reset using the *remove all* command or the R line command.



Note: Elementary items are modifiable from within the keep window.

In the following panel the program has stopped at a breakpoint. To display all the data items in the MOVE statement, you must type a K on each of the lines on which the statement occurs. In this case, you must type K on lines 13400, 13410, 13500, and 13600. Pressing Enter causes the command to process.

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
        7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500 GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
0001      012800 MOVE HD1 TO LINE-OUT.
0001      012900 PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
0001      013200 MOVE MASK1 TO BIN1.
0001      013300 MOVE ZERO TO MASK1.
K-->      013400 MOVE MASK1 TO BIN1
K          013410 BIN2
K          013500 BIN3
K          013600 BIN4.
-->      013700 TABLE-MOVE.
U-->      013736 MOVE TABLE-FILL TO WA-TABLE-1.
-->      013737 NUM-EDITED-TEST.

```

The following panel shows the keep window displays for data items MASK1, BIN1, BIN2, BIN3, and BIN4:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
-----
00204DA0 AN      004000 03 MASK1          00
00204DB2 NB-S    004600 03 BIN1          +00000001.
00204DB6 NB-S    004700 03 BIN2          +00000016.
00204DBA NB-S    004800 03 BIN3          +00000032.
00204DBE NB-S    004900 03 BIN4          +00000077.
-----
        7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500 GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
0001      012800 MOVE HD1 TO LINE-OUT.
0001      012900 PERFORM WRITE-LINE.
U0001      013100 FORCING-THE-ABEND.
0001      013200 MOVE MASK1 TO BIN1.
0001      013300 MOVE ZERO TO MASK1.
U-->      013400 MOVE MASK1 TO BIN1
          013410 BIN2

```

O or X Command (Turn Off Breakpoints)

The O or X line command removes a breakpoint that is set for the line on which it is typed. A blank in the first field on a line also resets the breakpoint. The O line command should only be used on lines that contain a U,), or S in the first column (where a breakpoint is set). An O on any other line has no effect. The number of O commands that may be processed is limited only by the display screen size.

C, W, and V line commands are not displayed and therefore cannot be removed with an O or X.



Note: X is a synonym of the R line command when it is used in the keep window. When it is used on a source line, X is a synonym of the O line command.

In the following panel, you have entered an O line command on lines 13400 and 13736 by typing over the U's in the first column:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
        7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500      GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700      OPEN OUTPUT OUTPUT01.
0001      012800      MOVE HD1 TO LINE-OUT.
0001      012900      PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
0001      013200      MOVE MASK1  TO BIN1.
0001      013300      MOVE ZERO   TO MASK1.
o--->      013400      MOVE MASK1  TO BIN1
          013410          BIN2
          013500          BIN3
          013600          BIN4.
--->      013700 TABLE-MOVE.
o--->      013736      MOVE TABLE-FILL TO WA-TABLE-1.
--->      013737 NUM-EDITED-TEST.

```

After you press Enter, the U's are no longer on the lines. This indicates that the breakpoints have been reset as shown in the following panel:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
        7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500      GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700      OPEN OUTPUT OUTPUT01.
0001      012800      MOVE HD1 TO LINE-OUT.
0001      012900      PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
0001      013200      MOVE MASK1  TO BIN1.
0001      013300      MOVE ZERO   TO MASK1.
--->      013400      MOVE MASK1  TO BIN1
          013410          BIN2
          013500          BIN3
          013600          BIN4.
--->      013700 TABLE-MOVE.
--->      013736      MOVE TABLE-FILL TO WA-TABLE-1.
--->      013737 NUM-EDITED-TEST.

```

You can confirm that you reset the breakpoints by using the LISTAT command to produce the Breakpoint Status panel.

P Command (Print Display)

For COBOL or PL/I programs, the P line command writes the display in the session log for all the data items on the line where you typed the command. For assembler programs, the P line command prints the operands for the current instruction in register notation. The data is written as it would be displayed using the D line command. For a description of the Display Panel, see [Data Display Panels in Debugging Panels \(see page 19\)](#).

You can type the P line command on any line in the program listing. You can enter multiple P commands on different lines.

For further description and a sample of the session log, see [Session Log Facility \(Review Debugging Session\)](#) at the end of this section.

R Command (Reset Data Item Display)

The R line command resets a data item display from an LDI command, an L line command, an LDX command, or an H line command. Place the R line command on a line on the program listing to reset all the data items on that line for which List Data Item displays are present, or place it directly on a listed data item display to reset that display.



Note: X is a synonym of the R line command when it is used in the keep window. When it is used on a source line, X is a synonym of the O line command.

In the following panel, an LDI has been done for MASK1 and BIN1, and an LDX has been done for BIN2. By placing an R on the data item display for BIN1 and line 13410 containing BIN2, these data items are reset after you press Enter.

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>                                           SCROLL==> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
-----
r 00204DA0 AN      004000 03 MASK1                      00
  00204DB2 NB-S    004600 03 BIN1                      +00000001.
  00204DB6 NB-S    004700 03 BIN2                      +00000016.
                                DATA OCCUPIES 0004 BYTES OF STORAGE      ZONES 0001
                                NUMERICS 0000
-----
      7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500 GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
0001      012800 MOVE HD1 TO LINE-OUT.
0001      012900 PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
0001      013200 MOVE MASK1 TO BIN1.
0001      013300 MOVE ZERO TO MASK1.

--->      013400 MOVE MASK1 TO BIN1
r          013410 BIN2

```

The following panel shows the results of the R line commands. The BIN1 and BIN2 displays have been reset, leaving only the display for MASK1:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
COMMAND ==>                                     SCROLL==> PAGE
TRACE=> 013400 013300 013200 013100 020100 020000 012900 012800 012700 012600
-----
00204DA0 AN      004000 03 MASK1                                00
-----
      7          CAMRCOB2          15.49.31          JUN 30,2014
0001      012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      012400 DEMO-BEGIN.
0001      012500      GO TO DEMO-INIT.
0001      012600 DEMO-INIT.
0001      012700 OPEN OUTPUT OUTPUT01.
0001      012800 MOVE HD1 TO LINE-OUT.
0001      012900 PERFORM WRITE-LINE.

U0001      013100 FORCING-THE-ABEND.
0001      013200 MOVE MASK1 TO BIN1.
0001      013300 MOVE ZERO TO MASK1.

--->      013400 MOVE MASK1 TO BIN1
          013410 BIN2
          013500 BIN3
          013600 BIN4.
--->      013700 TABLE-MOVE.
--->      013736 MOVE TABLE-FILL TO WA-TABLE-1.

```

S Command (Skip this Statement)

The S (Skip) line command when placed directly next to a statement causes the statement to be skipped when the program is executing. The S remains on the line to show you which statements are not executed. Skip is considered an unconditional breakpoint.

You may type other commands over the S. To cancel a Skip line command use a blank, X, or O line command. If you use OFF ALL to remove all breakpoints, all skip conditions are also removed.



Important! When a program has been optimized, or when the statement being skipped is any of the following, the S (Skip) line command may have undesired results:

The last statement in the program

- The last statement in a performed paragraph
- Immediately followed by an ELSE statement

For more details, see the [SKIP Command \(Skip a Statement\)](#) (see page 103) .

The following panel shows a skip breakpoint being set at statement 21010. The S causes the call to the program A1 to be skipped as the program is executed:

```

CAMRCOB2 ----- CA InterTest Batch *INITIAL* Intercept -----
COMMAND ==>                                     SCROLL==> HALF
TRACE=> 020400
      8          17.28.55          MAR 10,2014
*- -> 020400 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-

```

```

*-->      020500 DEMO-BEGIN.
*-->      020600      GO TO DEMO-INIT.
*-->      020700 DEMO-INIT.
*-->      020800      OPEN OUTPUT OUTPUT01.
*-->      020900      MOVE HD1 TO LINE-OUT.
*-->      021000      PERFORM WRITE-LINE.
S*-->      021010      CALL 'A1'.
*-->      021020      CALL 'A2'.

U*-->      021200 FORCING-THE-ABEND.
*-->      021300      MOVE MASK1 TO BIN1.
*-->      021310      DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
*-->      021400      MOVE ZERO TO MASK1.
*-->      021500      MOVE MASK1 TO BIN1
*-->      021600                      BIN2
*-->      021700                      BIN3
*-->      021800                      BIN4.
*-->      021900      IF BIN1 = BIN2
*-->      022000      NEXT SENTENCE.

```

The following panel shows what the program looks like after you have used the GO command to resume program execution. The program is intercepted by the breakpoint on line 21200. Note that there is no frequency counter on statement 21010. This statement was not executed.

```

CAMRCOB2 ----- CA InterTest Batch UNCOND BEFORE Intercept -----
COMMAND ==>                                     SCROLL==> HALF
TRACE=> 021200 021020 021000 020900 020800 020700 020600 020500 020400
      8                                     17.28.55      MAR 10,2014
0001      020400 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-
0001      020500 DEMO-BEGIN.
0001      020600      GO TO DEMO-INIT.
0001      020700 DEMO-INIT.
0001      020800      OPEN OUTPUT OUTPUT01.
0001      020900      MOVE HD1 TO LINE-OUT.
0001      021000      PERFORM WRITE-LINE.
S---->      021010      CALL 'A1'.
0001      021020      CALL 'A2'.

U---->      021200 FORCING-THE-ABEND.
---->      021300      MOVE MASK1 TO BIN1.
---->      021310      DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
---->      021400      MOVE ZERO TO MASK1.
---->      021500      MOVE MASK1 TO BIN1
---->      021600                      BIN2
---->      021700                      BIN3
---->      021800                      BIN4.
---->      021900      IF BIN1 = BIN2
---->      022000      NEXT SENTENCE.

```

V Command (Set Variable Change Breakpoint)

The V line command sets a variable change breakpoint at the specified line. A when-name is automatically generated. The name is VRnnnnnn where n=line number of the breakpoint. This means you can have only one conditional set by the V line command per line. The breakpoint is triggered before the line is executed when the application determines that the value of a variable in the line changes. The condition is *not* checked before each verb. This command is only valid for COBOL or PL/I programs.

Cursor-sensitive variable change breakpoint: If you enter V and the cursor is placed on a COBOL or PL/I data item, pressing Enter designates that item as the variable change item.

The following panel shows variable change breakpoints being set on lines 1149 and 1154 of the program. You must press Enter after entering the breakpoints in order to set them:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----
Command ==> Scroll==> CUR
TRACE=> 001151 001150 001149 001148 001162 001155 001154 001153 001151 001150 0
U0001 001141 ORDER-CALC.
0001 001142 MOVE CUSTOMER-VALUES TO CUSTOMER-ORDER.
0001 001143 MOVE ZEROES TO ORDER-SUB-TOTAL, SUB-6, LOOP-0
0001 001144 PERFORM TOTAL-ORDER THRU TOTAL-ORDER-EX
001145 UNTIL SUB-6 EQUAL 5.
U---> 001146 ORDER-CALC-EX.
---> 001147 GO TO OPTIONS.
0101 001148 TOTAL-ORDER.
v0101 001149 ADD 1 TO SUB-6.
0101 001150 ADD 1 TO LOOP-OUT.
0100 001151 MULTIPLY UNIT-PRICE(SUB-6) BY NO-OF-WIDGETS(S
001152 GIVING ITEM-TOTAL.
0100 001153 ADD ITEM-TOTAL TO ORDER-SUB-TOTAL.
v0100 001154 MOVE 1 TO SUB-6.
PP 5668-958 IBM VS COBOL II Release 3.2 09/05/90 CAMRCOB2
LineID PL SL ----+*A-1-B--+----2----+----3----+----4----+----5----+

```

) Command (Set Unconditional "After" Breakpoints)

The) line command performs the same function as the AT or UNCOND command. An unconditional "after" breakpoint is set on the line where the A or U is typed. The program stops *after* executing the statement. A) displays on the line to show you where breakpoints have been set. The) line command is only valid on executable lines of the program listing. Executable lines contain a frequency counter number or arrow. There is no limit other than the size of the screen. If you press Enter or the scroll keys, it sets the breakpoints.

You can type other commands over the). Reset breakpoints using the OFF, O, X, or blank line commands.

The following panel shows an "after" breakpoint being set on line 13736 of the program:

```

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----

COMMAND ==> SCROLL==> PAGE

TRACE= 012400 012300

          7          CAMRCOB2          15.49.31          JUN 30,2014

0001          012300 PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-

--->          012400 DEMO-BEGIN.

--->          012500          GO TO DEMO-INIT.

--->          012600 DEMO-INIT.

--->          012700          OPEN OUTPUT OUTPUT01.

--->          012800          MOVE HD1 TO LINE-OUT.

```

```

--->      012900      PERFORM WRITE-LINE.

u--->      013100  FORCING-THE-ABEND.

--->      013200      MOVE MASK1  TO BIN1.

--->      013300      MOVE ZERO   TO MASK1.

u--->      013400      MOVE MASK1  TO BIN1

              013410                      BIN2

              013500                      BIN3

              013600                      BIN4.

--->      013700  TABLE-MOVE.

)--->      013736      MOVE TABLE-FILL TO WA-TABLE-1.

--->      013737  NUM-EDITED-TEST.

--->      013738      MOVE IN-FUNC TO OUT-FUNC.

--->      013739      MOVE IN-FUNC TO OUT-FUNC2.

```

Press Enter to set the breakpoints.

The listing appears as shown in the following panel. The line that has an "after" breakpoint set displays a) in the first column to show you where it is.

CAMRCOB2 ----- CA InterTest Batch NEXT BEFORE Intercept -----

COMMAND ==>

SCROLL==> PAGE

TRACE=> 012400 012300

```

          7          CAMRCOB2          15.49.31          JUN 30,2014

0001      012300  PROCEDURE DIVISION USING PARMs TEST-THREE TEST-FOUR PASS-

--->      012400  DEMO-BEGIN.

--->      012500      GO TO DEMO-INIT.

--->      012600  DEMO-INIT.

```

```

--->      012700      OPEN OUTPUT OUTPUT01.

--->      012800      MOVE HD1 TO LINE-OUT.

--->      012900      PERFORM WRITE-LINE.


U--->      013100 FORCING-THE-ABEND.

--->      013200      MOVE MASK1 TO BIN1.


--->      013300      MOVE ZERO TO MASK1.

U--->      013400      MOVE MASK1 TO BIN1

                013410                        BIN2

                013500                        BIN3

                013600                        BIN4.

--->      013700 TABLE-MOVE.

)--->      013736      MOVE TABLE-FILL TO WA-TABLE-1.

--->      013737 NUM-EDITED-TEST.

--->      013738      MOVE IN-FUNC TO OUT-FUNC.

--->      013739      MOVE IN-FUNC TO OUT-FUNC2.

```

You can also use the LISTAT command to show the Breakpoints panel as follows:

```

----- CA InterTest Batch BREAKPOINTS PANEL -----

COMMAND ==>                                SCROLL ==> CUR

***** Top of Data *****

UNCONDITIONAL BREAKPOINTS FOLLOW:

```

```

PROGRAM   STMT#   COUNT  EXEC  B/A  COMMANDS TO BE EXECUTED
-----
.  CAMRCOB2  000589  0000   0000   B

SOURCE LINE: SPLIT-SCREEN.

.  CAMRCOB2  000541  0000   0000   B

SOURCE LINE: PAY-CALC.

.  CAMRCOB2  000659  0000   0000   A

SOURCE LINE: ORDER-CALC.

***** BOTTOM OF DATA *****

```

+ (plus) Command (Increment a Subscript)

Use the plus (+) line command to increment the value of a subscript on a displayed data item. Display the data item using the LDI or LDX control commands or the L or H line commands. The format of the command is `+nnn` where `nnn` is any number from 1 to 999. The subscript is incremented by the amount specified by `nnn`. The default is 1, and this is the value that is used if `nnn` is not specified. More than one subscript may be incremented or decremented at the same time. This command is only valid for COBOL or PL/I programs.

The following panel shows a data display for WA-T1-3A(BIN1,4,1) of WA-TABLE-1 of WA-LEVEL1. Currently, WA-T1-3A(2,4,1) is displayed. Note the address and the value. The first subscript (BIN1) and the third subscript are incremented by the value of 2.

```

CAMRCOB2 ----- CA InterTest Batch UNCOND BEFORE Intercept -----
COMMAND ==>                                     SCROLL==> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
-----
+2      00262E08 ND      016800 06 WA-T1-3A      01.
      BIN1 (2)
      (4)
+2      (1)
      OF WA-TABLE-1
      OF WA-LEVEL1
-----

0001      021200 FORCING-THE-ABEND.
0001      021300 MOVE MASK1 TO BIN1.
0001      021310 DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
0001      021400 MOVE ZERO TO MASK1.
0001      021500 MOVE MASK1 TO BIN1
      021600 BIN2
      021700 BIN3
      021800 BIN4.
0001      021900 IF BIN1 = BIN2

```

```

0001      022000      NEXT SENTENCE.
0001      022100 TABLE-MOVE.
0001      022200      MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U---->    022300 NUM-EDITED-TEST.
---->     022310      MOVE 'XP' TO ED-ALPHA.

```

Pressing Enter causes the line commands to be executed. The value of BIN1 and the value of the third subscript have been incremented by 2, so that WA-T1-3A(4,4,3) is now the table item displayed. Note that by incrementing the subscript variable BIN1, the value of BIN1 has been changed to 4 as shown in the following panel:

```

CAMRCOB2 ----- CA InterTest Batch UNCOND BEFORE Intercept -----
COMMAND ==>                                           SCROLL==> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
-----
00262E6C ND      016800 06 WA-T1-3A                      03.
                        BIN1 (4)
                        (4)
                        (3)
                        OF WA-TABLE-1
                        OF WA-LEVEL1
-----

0001      021200 FORCING-THE-ABEND.
0001      021300      MOVE MASK1 TO BIN1.
0001      021310 DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
0001      021400      MOVE ZERO TO MASK1.
0001      021500      MOVE MASK1 TO BIN1
                        021600          BIN2
                        021700          BIN3
                        021800          BIN4.
0001      021900      IF BIN1 = BIN2
                        022000      NEXT SENTENCE.
0001      022100 TABLE-MOVE.
0001      022200      MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U---->    022300 NUM-EDITED-TEST.
---->     022310      MOVE 'XP' TO ED-ALPHA.

```

- (minus) Command (Decrement a Subscript)

Use the minus (-) command to decrement the value of a subscript on a displayed data item. Display the data item using the LDI or LDX control commands or the L or H line commands. The format of the command is *-nnn* where *nnn* is any number from 1 to 999. The subscript is decremented by the amount specified by *nnn*. The default is 1, and this is the value that is used if *nnn* is not specified. You may increment or decrement more than one subscript at the same time. This command is only valid for COBOL or PL/I programs.

The following panel shows a data display for WA-T1-3A(BIN1,4,3) of WA-TABLE-1 of WA-LEVEL1. Currently, WA-T1-3A(4,4,3) is displayed. Note the address and the value. The first subscript (BIN1) is decremented by 3, and the second and third subscripts are decremented by 1:

```

CAMRCOB2 ----- CA InterTest Batch UNCOND BEFORE Intercept -----
COMMAND ==>                                           SCROLL==> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
-----
-3 00262E6C ND      016800 06 WA-T1-3A                      03.
-1      BIN1 (4)
-1      (4)
      (3)
      OF WA-TABLE-1
      OF WA-LEVEL1
-----

0001      021200 FORCING-THE-ABEND.
0001      021300      MOVE MASK1 TO BIN1.

```



```

0001      021310      DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
0001      021400      MOVE ZERO      TO MASK1.
0001      021500      MOVE MASK1     TO BIN1
                                BIN2
                                BIN3
                                BIN4.
0001      021900      IF BIN1 = BIN2
                                NEXT SENTENCE.
0001      022100      TABLE-MOVE.
0001      022200      MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U---->    022300      NUM-EDITED-TEST.
---->    022310      MOVE 'XP' TO ED-ALPHA.

```

Pressing Enter causes the line commands to be executed. The value of BIN1 has been decremented by 3, and the second and third subscripts have been decremented by 1, so that WA-T1-3A(1,3,2) is now the table item displayed.

Note that by decrementing the subscript variable BIN1, the value of BIN1 has been changed to 1 as shown in the following panel:

```

CAMRCOB2 ----- CA InterTest Batch UNCOND BEFORE Intercept -----
COMMAND ==>                                           SCROLL==> HALF
TRACE=> 022300 022200 022100 021900 021500 021400 021310 021300 021200 021020
-----
00262DCF ND      016800 06 WA-T1-3A                                02.
                                BIN1 (1)
                                (3)
                                (2)
                                OF WA-TABLE-1
                                OF WA-LEVEL1
-----

0001      021200      FORCING-THE-ABEND.
0001      021300      MOVE MASK1     TO BIN1.
0001      021310      DISPLAY MASK1 BIN1 BIN2 BIN3 BIN4.
0001      021400      MOVE ZERO      TO MASK1.
0001      021500      MOVE MASK1     TO BIN1
                                BIN2
                                BIN3
                                BIN4.
0001      021900      IF BIN1 = BIN2
                                NEXT SENTENCE.
0001      022100      TABLE-MOVE.
0001      022200      MOVE TABLE-FILL TO WA-TABLE-1 OF WA-LEVEL1.
U0001     022300      NUM-EDITED-TEST.

```

< nnnn (Scroll Left) Command

The scroll left (<) command is used to scroll data in the keep window left. The format of the command is <[nnnn] where *nnnn* is any number from 1 to 9999. If the data length is greater than the display area, the data is scrolled the number of bytes indicated by *nnnn*. Data cannot be scrolled beyond the length of the data item. If the *nnnn* value is greater than the length of the data item, the display is positioned at the last *n* number of bytes that fit in the display area.

The following panel shows a data item scrolled left 5 bytes:

```

CAMRCOB2 ----- CA InterTest Batch *INITIAL* Intercept -----
COMMAND ==>                                           SCROLL ==> CSR
TRACE=>
-----
1F0A9630      000838 01 TRL-TABLE
1F0A9635 AN    000839 03 FILLER                                (+0005) EFGHIJKLMNOPQRSTUVWXYZ
1F0A964B AN    000841 03 FILLER                                1234567890
1F0A9655 AN    000843 03 FILLER                                -+.,;:;#*/()@&%$¢?
-----
000875      PROCEDURE DIVISION USING PARAMETER-AREA.

```

```

*--> 000876      START-PROGRAM.
*--> 000877      OPEN OUTPUT REPORT-OUT.
000878      * -----*
000879      * Welcome to the CA InterTest Batch COBOL II      *
000880      * Demonstration Program. You are now at the      *
000881      * Initial Breakpoint panel which is displayed    *
000882      * upon entry to the first program to be tested.  *
000883      * At this point, other breakpoints can be set    *
000884      * and control commands can be issued before the  *
000885      * program is executed.                            *
000886      * -----*
*--> 000887      MOVE R1498-TIMEI TO R1498-TIME.
*--> 000888      PERFORM TIME-EDIT THRU TIME-EDIT-EX.
000889      *
*--> 000890      MOVE R1498-TIME0 TO R1498-TIME.
*--> 000891      PERFORM TIME-EDIT THRU TIME-EDIT-EX.
*--> 000892      GO TO DISPLAY-1ST-PANEL.
Note: Scrolling starts from relative zero. In the previous example the displayed character "E" is at real position 6 (+0005 relative to 0).

```

> nnnn (Scroll Right) Command

The scroll right (>) command is used to scroll data in the keep window right. The format of the command is >[nnnn] where *nnnn* is any number from 1 to 9999. If the data length is greater than the display area, the data is scrolled the number of bytes indicated by *nnnn*. Data cannot be scrolled beyond the length of the data item. If the *nnnn* value is greater than the length of the data item, the display is positioned at the first *n* number of bytes that fit in the display area.

The following panel shows a data item scrolled right 3 bytes:

```

CAMRCOB2 ----- CA InterTest Batch *INITIAL* Intercept -----
COMMAND ==>                                     SCROLL ==> CSR
TRACE=>
-----
1F0A9630      000838 01 TRL-TABLE
1F0A9632 AN    000839 03 FILLER                      (+0002) BCDEFGHIJKLMNOPQRS
1F0A964B AN    000841 03 FILLER                      1234567890
1F0A9655 AN    000843 03 FILLER                      -+=.,;:##*/()@&%$¢?
-----
000875      PROCEDURE DIVISION USING PARAMETER-AREA.
*--> 000876      START-PROGRAM.
*--> 000877      OPEN OUTPUT REPORT-OUT.
000878      * -----*
000879      * Welcome to the CA InterTest Batch COBOL II      *
000880      * Demonstration Program. You are now at the      *
000881      * Initial Breakpoint panel which is displayed    *
000882      * upon entry to the first program to be tested.  *
000883      * At this point, other breakpoints can be set    *
000884      * and control commands can be issued before the  *
000885      * program is executed.                            *
000886      * -----*
*--> 000887      MOVE R1498-TIMEI TO R1498-TIME.
*--> 000888      PERFORM TIME-EDIT THRU TIME-EDIT-EX.
000889      *
*--> 000890      MOVE R1498-TIME0 TO R1498-TIME.
*--> 000891      PERFORM TIME-EDIT THRU TIME-EDIT-EX.
*--> 000892      GO TO DISPLAY-1ST-PANEL.

```



Note: Scrolling starts from relative zero. In the previous example the displayed character "B" is at real position 3 (+0002 relative to 0)

Control Commands

Control commands control the test session. Enter them on the command line of the Intercept panel. Enter multiple control commands on the command line by separating each command with a semicolon (;).

- [ADVANCE Command \(Follow Trace Pointers Forward\) \(see page 70\)](#)
- [AT \(or UNCOND\) Command \(Set an Unconditional Breakpoint\) \(see page 71\)](#)
- [CORE Command \(Display Memory\) \(see page 73\)](#)
- [COUNTS Command \(Control Frequency Counting\) \(see page 73\)](#)
- [CS Command \(Current Statement\) \(see page 74\)](#)
- [DATAMON \(Enable Data Monitoring\) \(see page 74\)](#)
- [DDALLOC Command \(Allocate a DD\) \(see page 75\)](#)
- [DDFREE Command \(Free a DD\) \(see page 75\)](#)
- [DDQ Command \(Query a DD\) \(see page 75\)](#)
- [DISPLAY Command \(Format Data or the COBOL Working Storage Section\) \(see page 76\)](#)
- [DROP Command \(DROP Symbolic Name\) \(see page 77\)](#)
- [DROPUSE Command \(Remove a Using\) \(see page 77\)](#)
- [DUMP Command \(Terminate Testing with a Dump\) \(see page 78\)](#)
- [EQUATE Command \(Specify Symbolic Name\) \(see page 79\)](#)
- [EXSUM Command \(Display Execution Summary\) \(see page 80\)](#)
- [FILES Command \(Display COBOL FD Status\) \(see page 81\)](#)
- [FM Command \(Invoke CA File Master Plus\) \(see page 81\)](#)
- [FREQ Command \(Controls FREQ Counter\) \(see page 81\)](#)
- [GO Command \(Continue Test Session\) \(see page 82\)](#)
- [INCLUDE Command \(Execute Application Commands\) \(see page 83\)](#)
- [LDA \(or AUTOKEEP\) Command \(List Data Automatic\) \(see page 84\)](#)
- [LDI \(or KEEP\) Command \(List Data Item\) \(see page 85\)](#)
- [LDX \(or KEEPX\) Command \(List Data Item in Hex\) \(see page 87\)](#)
- [LEQ Command \(List Equated Names\) \(see page 89\)](#)
- [LINKAGE Command \(Format the COBOL Linkage Section\) \(see page 89\)](#)
- [LISTAT Command \(List Breakpoints\) \(see page 90\)](#)
- [LISTBP Command \(List All Breakpoints\) \(see page 90\)](#)
- [LISTLBL Command \(List Labels\) \(see page 90\)](#)
- [LISTUSE Command \(List Usings\) \(see page 90\)](#)
- [LISTWHEN Command \(List WHEN Conditions\) \(see page 91\)](#)
- [MAP Command \(Region MAP Display\) \(see page 91\)](#)
- [MLOG Command \(Record Debugging Session\) \(see page 91\)](#)
- [NEXT Command \(Executing Verbs\) \(see page 92\)](#)
- [NOFREQ Command \(Remove Frequency Counters\) \(see page 93\)](#)
- [OFF \(or OFFU\) Command \(Remove Unconditional Breakpoints\) \(see page 93\)](#)
- [OFFWN \(or OFFC\) Command \(Remove Conditional Breakpoints\) \(see page 95\)](#)
- [POINT Command \(Direct PREV and ADVANCE Pointer\) \(see page 95\)](#)
- [PREV Command \(Trace in Reverse\) \(see page 96\)](#)

- PS Command (Print Stream) (see page 97)
- QUALIFY Command (Set Current Program Id) (see page 97)
- QUIT Command (Terminate Session) (see page 98)
- RDI or REMOVE Command (Reset Data Item) (see page 98)
- RECORDS Command (Show COBOL Record Formats) (see page 99)
- REFRESH Command (see page 100)
- REGS Command (see page 100)
- RESET Command (see page 100)
- RESTART Command (see page 101)
- RUN Command (Resume Execution) (see page 101)
- SDWA Command (System Diagnostic Work Area) (see page 101)
- SET Command (Change Data Item Value) (see page 102)
- SKIP Command (Skip a Statement) (see page 103)
- SLOW (or AUTOSTEP) Command (Slowly Resume Program Execution) (see page 105)
- SNAP Command (Produce a SNAP Report) (see page 105)
- SPEED Command (Accelerate Program Testing) (see page 106)
- STEP Command (Set Count for GO Command) (see page 106)
- SUSPEND Command (Suspend a Batch Link Session) (see page 107)
- TRACE SOURCE Command (Trace with Source Code Displayed) (see page 108)
- TRACE Command (Controls Program Trace Entries) (see page 108)
- TRP Command (Trace Paragraphs) (see page 109)
- USING Command (Assign a Register or Address to a DSECT or CSECT) (see page 109)
- WHEN (or COND) Command (Set Conditional Breakpoints) (see page 109)
- * Command (Add Comments in Session Log) (see page 112)
- /clist Command (Execute CAMRCMD) (see page 112)

Assembler Indirect Addressing Note: Some control commands can accept an expression in *register* notation when debugging an assembler program. The format of an expression is as follows:

<base <suffix>> <<offset<suffix>>...>

Operands	Description
base	hexadecimal address, or register number (that is, R1 for general register 1), or a data name
suffix	% for indirect addressing in 24 bit mode ? for indirect addressing in 31 bit mode ! for indirect addressing in 64 bit mode
offset	plus (+) or minus (-) a hexadecimal value up to 4 characters long



Note: When starting an expression with a hexadecimal address, it must start with a plus (+) sign.

The following table shows a list of control commands available when using your product:

Command	Definition
ADVANCE	Follows the trace pointers forward.
AT	Sets an unconditional breakpoint (same as UNCOND).
CORE	Displays memory for the debugging region.
COUNTS	Enables frequency counting, reset counters, display counters.
CS	Highlights the current statement.
DATAMON	Enables the data monitoring feature.
DDALLOC	Allocates a DD.
DDFREE	Frees a DD.
DDQ	Queries a DD.
DISPLAY	Formats a data item or the COBOL Working Storage section.
DROP	Drops the symbolic name.
DROPUSE	Removes a using that was set with the USING command.
DUMP	Terminates testing with a dump.
EQUATE	Specifies symbolic name.
EXSUM	Provides the execution summary display.
FILES	Displays COBOL FD status.
FM	Starts CA File Master Plus.
FREQ	Controls verb execution counts.
GO	Resumes execution of the application.
INCLUDE	Executes a set of application commands.
LDA (or AUTOKEEP)	Displays all data items referenced by the current statement.
LDI (or KEEP)	Lists data item.
LDX (or KEEPX)	Lists data item in hexadecimal.
LEQ	Lists equated symbolic names.
LINKAGE	Formats the COBOL LINKAGE section.
LISTAT	Lists breakpoints.
LISTBP	Displays the BREAKPOINT panel.
LISTLBL	Lists labels added with the .label command.
LISTUSE	Displays the USING STATUS panel.
LISTWHEN	Lists WHEN conditions.
MAP	Invokes the Region Map Display panel.
MLOG	Records a debugging session.
NEXT	Executes the next <i>n</i> verbs.
NOFREQ	Removes frequency counters.

Command	Definition
OFF (or OFFU)	Removes unconditional breakpoints.
OFFWN (or OFFC)	Removes conditional breakpoints.
POINT	Directs PREVIOUS and ADVANCE pointer.
PREV	Follows trace pointers in reverse.
PS	Prints stream to session log.
QUALIFY	Sets current program id.
QUIT	Terminates the test session.
RDI	Resets data item (or REMOVE).
RECORDS	Formats data in the COBOL FILE section.
REFRESH	Displays updated split screen values.
REGS	Displays registers for assembler programs.
RESET	Removes column numbers from display.
RESTART	Restarts a batch link debugging step.
RUN	Removes breakpoints and execute.
SDWA	Displays the system diagnostic work area.
SET	Changes value of a data item.
SKIP	Skips a statement.
SLOW	Observes execution a verb at a time.
SNAP	Produces a CA Optimizer, CA Optimizer/II, or CA SymDump Batch SNAP report.
SPEED	Accelerates program testing.
STEP	Sets count for GO command.
SUSPEND	Detaches a terminal from a batch link debugging session without terminating the session.
TRACE	Controls/displays program trace entries.
TRACE SOURCE	Controls/displays program trace entries and their corresponding source code.
TRP	Displays traced paragraph entries.
UNCOND	Sets an unconditional breakpoint (same as AT).
USING	Sets an assembler base register for a DSECT that currently has no active USING.
WHEN (or COND)	Sets conditional breakpoints (or COND).
*	Adds a comment to the session log.
/clist	Executes CLIST CAMRCMD, called with parameter <i>clist</i> .

ADVANCE Command (Follow Trace Pointers Forward)

Referencing the trace table, the ADVANCE command highlights and displays the next statement that was executed. This feature lets the programmer visually follow in a forward sequence the statements that have been executed. This command is used along with the PREVIOUS command.

Syntax

Use the following syntax for ADVANCE:

```
ADVANCE <nnn|P nn>
```

You may enter this command in its abbreviated form: A.

Variables

Use the following variables with the ADVANCE command:

nnn	Specifies the number of executed statements for the session to move forward. The default is one.
P	Specifies that the next executed paragraph, procedure, or label is displayed.
nn	Specifies the number of paragraphs, procedures, or labels for the session to move forward. The default is one.

Usage Notes

The trace facility must be turned on for the ADVANCE command to function. The trace facility is turned on by default when you first invoke the application. For further information on turning the trace facility on and off, see the TRACE command later in this section.

The ADVANCE command visually displays the forward execution of the program using the trace table entries. It does not actually execute the program.

The ADVANCE command does not affect the point at which a program resumes execution. Execution is resumed by issuing a GO command.

If you have not used the PREV command, the ADVANCE P command scrolls the program to the current statement, highlights it, and displays END OF TRACE TABLE in the upper right-hand corner. Use this to return the program to the current statement after scrolling through the listing. The control command CS also displays and highlights the current statement.

If data monitoring was in effect when the statement highlighted from the ADVANCE command was executed, the values displayed for your data items are from the time before that statement was executed.

AT (or UNCOND) Command (Set an Unconditional Breakpoint)

Use the AT command to set breakpoints from the Intercept panel or to display the Breakpoint panel. The UNCOND command is a synonym of AT.

If you enter the AT or UNCOND command with operands in the Command field of an Intercept panel, the breakpoint is set.

If you enter AT or UNCOND without operands, the Breakpoint panel is displayed.

```
AT <<statement-number|paragraph-name|procedure-name|label-name> <AFTER|BEFORE> <
(subcommand-list)>>
<<ALL|program-name> <ENTRY|EXIT|LABEL|DBCALL> <(subcommand-list)>>
```

Variables

See the following variables with the AT or UNCOND commands:

statement-number	Specifies a statement-number in the program where a breakpoint is set.
paragraph-name	Specifies a paragraph-name in the COBOL program where a breakpoint is set.
procedure-name	Specifies a procedure-name in the PL/I program where a breakpoint is set.
label-name	Specifies label-name in the assembler or PL/I program where a breakpoint is set.
ENTRY	For COBOL programs, specifies that a breakpoint occurs each time the currently qualified program begins execution. For PL/I and assembler programs, specifies that a breakpoint occurs the first time the currently qualified program is entered.
EXIT	Specifies that a breakpoint occurs each time the currently qualified COBOL program finishes execution.
LABEL	Specifies that a breakpoint occurs each time the currently qualified program executes a paragraph, procedure, or label.
DBCALL	Specifies that a breakpoint occurs each time the currently qualified program makes a call to a database, such as DB2 or IMS
ALL ENTRY	Specifies that a breakpoint occurs each time a monitored COBOL program in the run unit begins execution and the first time a monitored assembler or PL/I program in the run unit begins execution.
ALL EXIT	Specifies that a breakpoint occurs each time any monitored COBOL program in the run unit finishes execution.
ALL LABEL	Specifies that a breakpoint occurs each time a monitored program executes a label.
ALL DBCALL	Specifies that a breakpoint occurs each time a monitored program makes a call to a database, such as DB2 or IMS.
AFTER	Specifies that the breakpoint should be triggered after the statement is executed.
BEFORE	Specifies that the breakpoint should be triggered before the statement is executed. This is the default.
subcommand-list	A list of application commands, separated by semicolons, to be executed when the breakpoint is encountered during the test. When testing under the ISPF dialog manager, you may enter only one subcommand unless the ISPF command delimiter character is changed to something other than the semicolon. This can be done under option 0.1 of the ISPF main menu.

Usage Notes

Set a breakpoint by the paragraph name or by the line number of the paragraph. Sample statements to set breakpoints by paragraph name are shown next:

```
041400 0000-INITIALIZATION SECTION.
041500
041600 MOVE +20 TO VARIABLE-LENGTH-1.
041700 MOVE +25 TO VARIABLE-LENGTH-2.
041800 MOVE +30 TO VARIABLE-LENGTH-3.
```

Set a breakpoint on the first statement of the example by either of the following AT commands:

AT 41400 or AT 0000-INITIALIZATION

Set a breakpoint with a list of application commands to be executed at the breakpoint. The commands are executed one at a time until the program resumes execution or until a transfer of panels occurs, at which point the remaining commands are ignored. Commands should be in parentheses, separated by semicolons, as follows:

```
AT 41700 (SET VARIABLE-LENGTH-2 = 50;GO 41800)
```

In this example, the program stops before executing statement 41700, and VARIABLE-LENGTH-2 is set to 50. The GO 41800 causes the program to resume execution at statement 41800 so the contents of VARIABLE-LENGTH-2 are not changed to +25. You can use commands like these to temporarily modify the execution flow of the program.

ISPF Note

When running under ISPF, specify only one command in the subcommand list because of restricted use of the semicolon, unless the ISPF command delimiter character has been changed. You can do this in option 0.1 of the ISPF main menu.

CORE Command (Display Memory)

The CORE command invokes the Display-Alter Memory panel. This panel lets you modify data based on its address. For more information, see [Core Option \(see page 36\)](#) and [Map Option \(see page 38\)](#).

Syntax

Use the following syntax for CORE:

```
CORE <data-item|paragraph-name|address|expression>
```

Usage Notes

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, you will see a warning panel. The CORE command shows only the current values of data items and not previous values. Previous values of data items cannot be modified.

COUNTS Command (Control Frequency Counting)

Use the COUNTS command to enable or disable frequency counting, reset frequency counters, and control the display of the frequency counters.

When enabled, frequency counting causes the debugger to count the number of times each statement is executed. When the display is active, the frequency counters appear on the Intercept panel in the left margin.

Syntax

Use the following syntax for COUNTS:

```
COUNTS <ON|OFF> <SHOW|NOSHOW> <RESET|RESET ALL|RESET progid>
```

Variables

Use the following variables with the COUNTS command:

ON	Enables frequency counting
OFF	Disables frequency counting
SHOW	Causes the frequency counters to be displayed on the Intercept panel
NOSHOW	Suppresses the display of the frequency counters on the Intercept panel
RESET	Clears to zero all of the counters for the current program
RESET ALL	Clears to zero all of the counters for all of the programs
RESET progid	Clears to zero all of the counters for the specified program

Usage Notes

When specifying COUNTS ON or OFF, you may also specify SHOW or NOSHOW as a second argument.

If frequency counting is enabled but the display is disabled, counting continues even though no counts are displayed.

CS Command (Current Statement)

The Current Statement (CS) command displays and highlights the current statement. This is especially useful after scrolling through the listing and any time the next statement to be executed is no longer displayed. You can also change the current statement by a passing statement number.

Syntax

Use the following syntax for CS:

```
CS <statement-number|paragraph-name|procedure-name|label-name>
```

Usage Notes

When using CS to modify the flow of execution of a COBOL program, be careful to avoid entering a paragraph and bypassing the PERFORM statement. When the exit to the paragraph is reached, a USER 519 abend may be issued because the references to the calling statement are not defined.

When using CS to modify the flow of execution of a PL/I program, be careful to avoid entering a block (that is, procedure, BEGIN block, or ON-unit) other than at the start of the block, or exiting a block other than at the END statement for the block, as it causes unpredictable results.

DATAMON (Enable Data Monitoring)

The DATAMON command enables data monitoring for the specified COBOL program. When this feature is used in conjunction with the trace navigation commands (PREV, ADV, POINT), you see past data values when displaying data.

Syntax

Use the following syntax for DATAMON:

```
DATAMON ON|OFF <pgmid>
```

You can use this command in its abbreviated form: DM.

Usage Notes

Enabling DATAMON lets you view past data values when navigating the trace table. If DATAMON was in effect for the statement highlighted by the trace navigation command (PREV, ADV, POINT), the data values displayed in the KEEP and DISPLAY commands will be past data values captured from a point in time *before* a COBOL statement referencing them is executed. These values are protected as you may not be able to update data while viewing past data values. For this reason, the SET command is disabled in this situation. When using the CORE command, a warning panel is shown indicating that the current data is shown for viewing and updating, not the past data values. Also, use of the + and - line commands are disabled when incrementing or decrementing a subscript symbolically.

The DATAMON command can be CPU intensive and it requires a significant amount of storage. The DATAMON command is only valid for COBOL programs.

DDALLOC Command (Allocate a DD)

The DDALLOC command displays the DDALLOC panel. From this panel, you allocate a DD to DUMMY, a SYSOUT class, or a data set. Note that this feature works in both batch link and foreground testing. Foreground users can also split their screen and use Option 3, ALLOCATIONS, to perform their allocations.

Syntax

Use the following syntax for DDALLOC:

```
DDALLOC <ddname>
```

You can enter this command in its abbreviated form: DDA.

DDFREE Command (Free a DD)

The DDFREE command lets you free a specified DD.

Syntax

Use the following syntax for DDFREE:

```
DDFREE ddname
```

You can enter this command in its abbreviated form: DDF.

DDQ Command (Query a DD)

The DDQ command invokes the DDName Query panel. The panel displays information about the allocated DD's.

Syntax

Use the following syntax for DDQ:

```
DDQ <ddname>
```

DISPLAY Command (Format Data or the COBOL Working Storage Section)

The DISPLAY command invokes the Display panel. You can modify the data values on this display.

Syntax

Use the following syntax for DISPLAY:

```
DISPLAY <data-item|expression <data-item> ...> <PRINT>
```

You can enter this command in its abbreviated form: D.



Note: Data for variables are case-sensitive.

Variables

Use the following variables with the DISPLAY command:

data item Causes a display of the data item, any subordinate level data items, and their data values. You may enter more than one data item.

expression For assembler programs, enter an expression in register notation.

PRINT Causes the data displayed by the command to be written to the session log.

Usage Notes

Consider the following COBOL WORKING-STORAGE definition:

```
WORKING-STORAGE SECTION.
01 WORKING-STORAGE-AREA.
   05 CONVERSION-AREA.
      10 CONV-DATE      PIC 9(7) VALUE ZERO.
      10 CONV-TIME      PIC 9(6)V999 VALUE ZERO.
   05 SELECTION CRITERIA.
      10 SELECT-START    PIC S9(15)V999 COMP.
      10 SELECT-END      PIC S9(15)V999 COMP.
```

If you enter the DISPLAY command without operands, it formats and displays all of the previous data items and their data values.

The command D CONVERSION-AREA would format and display CONVERSION-AREA, CONV-DATE, and CONV-TIME.

The command D SELECT-START would format and display only the data item SELECT-START.



Note: If data monitoring was in effect, data item values displayed are those captured at a point in time before the COBOL statement, positioned by Prev or Adv command is executed.

If a data item is modified on the other side of a split screen, you can update the Display panel to reflect any changes made by typing **REFRESH** on the command line.



Note: The data-item parameter may be a qualified data item, as in the following examples:

COBOL: Use the command `D EMPLOYEE-NO IN MASTER-RECORD` to refer to the EMPLOYEE-NO in MASTER-RECORD if EMPLOYEE-NO is not unique.

PL/I: Use the command `D MASTER.RECORD.EMPLOYEE_NO` to refer to the EMPLOYEE_NO in MASTER_RECORD if EMPLOYEE_NO is not unique.

Use the command `D NEXT_ELEM_PTR->ELEMENT` to refer to the allocation of based variable ELEMENT that is pointed to by NEXT_ELEM_PTR.

For assembler or PL/I programs, you need a data-item or expression operand.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items are from the time before that statement was executed.

DROP Command (DROP Symbolic Name)

Use the DROP command to remove the symbolic name for a data item that has been set by the EQUATE command. View the symbolic names using the LEQ command or set by using the EQUATE command. DROP without operands causes the Equate panel to be displayed. [For more information on the EQUATE command, see Breakpoint Panels \(see page 26\).](#)

Syntax

Use the following syntax for DROP:

```
DROP <symbolic-name>
```

You can enter this command in its abbreviated form: DR.

Variables

Use the following variable with the DROP command:

symbolic-name User-defined symbolic name for a data item.

DROPUSE Command (Remove a Using)

The DROPUSE command removes a using that was set with the USING command. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax for DROPUSE:

DROPUSE dsect|csect

Variables

Use the following variables with the DROPUSE command:

dsect	Assembler DSECT that you wish to map
csect	Assembler CSECT that you wish to map

You can enter this command in its abbreviated form: DROPU.

DUMP Command (Terminate Testing with a Dump)

Use the DUMP command to terminate the testing session with a dump. This command only works when the application being tested has previously abended.

Syntax

Use the following syntax for DUMP:

DUMP

Usage Notes

To use the DUMP command, you must pre-allocate an MVS dump data set (SYSMDUMP). Allocate the dump data set from Option 3 (ISPF) - ALLOCATION, of the Primary Option Menu; from ISPF option 6 using the TSO allocate command; or by pre-allocating a dump data set in the CLIST. The following example is a sample allocation as it appears in a CLIST:

```
CONTROL NOMSG
FREE DD(SYSMDUMP)
DEL yourdsn
CONTROL MSG
ALLOC DD(SYSMDUMP) DSN(yourdsn) NEW SPACE(105) CYL
UNIT(unitname) VOLUME(volser) CATALOG
```

Following is a sample dump allocation for CA Roscoe users:

```
FREE SYSMDUMP
DELETEDSN SYSMDUMP
ALLOC SYSMDUMP dsn=SYSMDUMP
SPACE=(cyl, (50,10))
DISP=(NEW,catalog)
VOL=
UNIT=
```

You must print SYSMDUMPs using the AMDPRDMP utility.



Note: You can also use the DDALLOC command to allocate the SYSMDUMP DD to an existing DSN or a SYSOUT class.

EQUATE Command (Specify Symbolic Name)

Use the EQUATE command to equate a data item with a symbolic name. You may then refer to the data item by either the data item or the symbolic name. View the symbolic names using the LEQ command or delete by using the DROP command. The EQUATE command supports qualified data items. Equate without operands causes the Equate panel to display. [For more information on the EQUATE command, see Breakpoint Panels \(see page 26\).](#)

Syntax

Use the following syntax for EQUATE:

```
EQUATE <symbolic-name data-item>
```

You can enter this command in its abbreviated form: EQU.

Variables

Use the following variables with the EQUATE command:

symbolic-name	A user-defined symbolic name for a data item
data-item	An elementary data item or qualified data item

Data-item can be a qualified data item with 1 or 2 levels of qualification, as described in the following examples:

COBOL: The following command equates NAME-ONE OF EMP-NAME OF PAYROLL-REC to the symbolic name N1. You may now specify N1 as the data item for control commands such as LDI and SET.

```
EQUATE N1 NAME-ONE OF EMP-NAME OF PAYROLL-REC
```

This variable may also be specified by equating the following:

```
EQU NO NAME-ONE
EQU EN EMP-NAME
EQU PR PAYROLL-REC
```

NO OF EN OF PR used as the data item in a control command also refers to NAME-ONE OF EMP-NAME OF PAYROLL-REC.

PL/I: The following command equates PAYROLL_REC.EMP_NAME.NAME_ONE to the symbolic name N1. You may now specify N1 as the data item for control commands such as LDI and SET.

```
EQUATE N1 PAYROLL_REC.EMP_NAME.NAME-ONE
```

You may also specify this variable by equating the following:

```
EQU NO NAME_ONE
EQU EN EMP_NAME
EQU PR PAYROLL_REC
```

PR.EN.NO used as the data item in a control command also refers to PAYROLL_REC.
EMP_NAME.NAME_ONE.

The data item may be a fixed or variably subscripted data item or indexed data item. The following commands equate a variably subscripted data item to the symbolic name W1.

COBOL:

EQ W1 WA-TI-3A(S1,S2,S3)

PL/I:

EQ W1 WA_TI_3A(S1,S2,S3)

The data item can also be a qualified subscripted or indexed data item, as in the following examples:

COBOL: This command equates the symbolic name W1 to the fourth value in the non-unique table WA-1A in the unique table WA-TABLE-1:

EQU W1 WA-1A OF WA-TABLE-1(4)

PL/I: This command equates the symbolic name W1 to the fourth value in the non-unique table WA_1A in the unique table WA_TABLE_1:

EQU W1 WA_TABLE_1.WA-1A(4)

EXSUM Command (Display Execution Summary)

The EXSUM command provides an execution summary display of the current program. The display shows the number of executable statements, the number of statements currently executed during the test, and the percentage of statements currently executed during the test.



Note: Statements executed when COUNTS OFF is in effect are not counted by the EXSUM command and will not get included in the summary.

Syntax

Use the following syntax for EXSUM:

EXSUM

You can enter this command in its abbreviated form: EXS.

The following panel is a sample Execution Summary Display:

```
----- CA InterTest Batch Execution Summary Display -----
PROGRAM ID:  INSTST
Number of executable statements: 21
Number of statements currently executed during this test: 21
Percentage of statements currently executed during this test: 100
```




Note: If COUNTS NOSHOW is in effect, no statements are counted and the Execution Summary Display will show N/A for Not Applicable for all three counts.

FILES Command (Display COBOL FD Status)

The FILES command displays the File Status panel. The File Status panel indicates whether the data sets in the FD section are open or closed, and it gives the DSORG and DCB information for the files. For more information on File Status Panels, see [Debugging Panels \(see page 19\)](#).

Syntax

Use the following syntax for FILES:

FILES

You can enter this command in its abbreviated form: FI.

Usage Notes

IMS databases are not supported on this display. This command is not supported for assembler or PL/I programs.

FM Command (Invoke CA File Master Plus)

The FM command invokes CA File Master Plus, if you have that product installed. For details on using that product, see the *CA File Master Plus* documentation.

Syntax

Use the following syntax for FM:

FM



Note: This command is valid for both batch link and foreground testing.

FREQ Command (Controls FREQ Counter)

The frequency command controls the statement frequency counter, but the frequency command and the NOFREQ command control whether the counters are displayed.



Note: The FREQ command is replaced by the COUNTS command, which is described earlier in this section.

Syntax

Use the following syntax for FREQ:

FREQ <ON|OFF|RESET>

You can enter the FREQ command in its abbreviated form: FR.

Variables

Use the following variables with the FREQ command:

ON	Enables frequency counting. This is the default. This command causes the Intercept panel to shift to the right six characters and displays one of the following in the left margin:
nnnn	A four-digit integer indicating the number of times the statement has executed. If the counter exceeds 9999, the number is represented as follows:
	<i>nnnK</i> = <i>nnn</i> x 1 thousand (for example, 010K= 10 thousand)
	<i>nnnM</i> = <i>nnn</i> x 1 million (for example, 010M= 10 million)
	<i>nnnG</i> = <i>nnn</i> x 1 billion (for example, 010G= 10 billion)
	If your frequency count has exceeded 9999 and you wish to see an exact count, issue an I (Info) line command to the left of the line. The full count is displayed in the upper right-hand corner of the Intercept panel.
--->	This line contains the beginning of a valid statement, but it has not yet been executed.
Spaces	The line number is not the beginning of a statement.
OFF	Disables frequency counting.
RESET	Zeroes the frequency counters for the currently qualified program.

Usage Notes

When you initially invoke the application, the frequency counter is turned on but is not displayed. To enable the display, enter the FREQ command without operands. Subsequently, if the FREQ OFF command is issued, the counters are no longer maintained, but the residual counts remain on the screen. To remove the counters and arrows from the screen, enter the NOFREQ command.

The status of the frequency command is maintained over sessions.

GO Command (Continue Test Session)

The GO command causes the test session to continue until the next intercept occurs.

Syntax

Use the following syntax for GO:

GO <Statement-number|paragraph-name|procedure-name|label-name>

Variables

Use the following variables with the GO command:

statement-number	Specifies a statement-number in the program where the program resumes execution.
paragraph-name	Specifies a paragraph-name in the program where execution resumes.
procedure-name	Specifies a procedure-name in the PL/I program where execution resumes.
label-name	Specifies a label-name in the assembler or PL/I program where execution resumes.

Specify a statement number, paragraph, procedure, or label name as the execution start point. GO with no operands resumes execution at the next logical statement unless the program was interrupted because of an ABEND. In this case, the program resumes execution at the statement where the ABEND occurred.

The program executes until one of the following occurs:

- A breakpoint is encountered.
- An ABEND occurs in the program.
- The step count (specified by the STEP command) is exhausted.
- The user interrupts the test by pressing the ATTENTION key.
- The program runs to normal completion.

Usage Notes

When using GO to modify the flow of execution of a COBOL program, be careful to avoid entering a paragraph and bypassing the PERFORM statement. When you reach the exit to the paragraph, you may receive a USER 519 ABEND because the references to the calling statement are not defined.

When using GO to modify the flow of execution of a PL/I program, be careful to avoid entering a block (that is, procedure, BEGIN block, or ON-unit) other than at the start of the block, or exiting a block other than at the END statement for the block, as it will cause unpredictable results.

INCLUDE Command (Execute Application Commands)

The INCLUDE command reads and executes a set of application commands from a partitioned data set.

Syntax

Use the following syntax for INCLUDE:

```
INCLUDE member-name
```

Variables

Use the following variable with the INCLUDE command:

member-name	Specifies a valid member name of a partitioned data set (PDS) allocated to ddname INT1CLIB.
--------------------	---

Usage Notes

Before you enter the INCLUDE command, perform an allocation to ddname INT1CLIB. Allocate the partitioned data set with an LRECL=80 and a RECFM=FB. Create the members using a text editor.

Use the INCLUDE command to execute a set of application commands for regression testing or retesting as code is modified during testing. For example, use the following commands to test a tax computation routine:

```
SET EMPLOYEE-NAME = 'PIERCE, MICHAEL'
SET EMPLOYEE-NUMBER = 3938
SET EMPLOYEE-STATE = 'CA'
WHEN TOOHIGH EMPLOYEE-SALARY GT 50000.00
AT 0100-CALC-FICA
```



Note: The INCLUDE command ignores all commands in the input stream following a command that causes the program to resume.

For example, GO or NEXT cause the program to resume execution; DISPLAY or LISTAT cause a transfer of panels. If the input stream includes these commands, any command following them is ignored.

LDA (or AUTOKEEP) Command (List Data Automatic)

The LDA or AUTOKEEP (list data automatic) command displays in the keep window all the data items referenced by the current statement.

The keep window appears on the Intercept panel immediately above the first line of the program listing.

Syntax

Use the following syntax for LDA or AUTOKEEP:

```
LDA <ON|OFF>
```

You may enter this command in its abbreviated form: AUTOK.

Variables

Use the following variables with LDA or AUTOKEEP:

ON	Enables the automatic list data facility.
OFF	Disables the automatic list data facility.

Usage Notes

LDA does not inhibit your ability to use the LDI (KEEP) or LDIX (KEEP HEX) commands to add static entries to the keep window. While automatically added data items are added and removed at each breakpoint, a static data item continues to be displayed in the keep window until removed manually.

When AUTOKEEP is in effect for an assembler routine, the operands for the current instruction are added to the keep window in register notation.

COBOL programs can be compiled with the ARITH(EXTEND) parameter which will allow a numeric field to be greater than 18 bytes. These fields will be displayed with the message 'TOO BIG TO DISPLAY' in the value. To view or change the actual value of the field, use the CORE command.

LDI (or KEEP) Command (List Data Item)

The list data item (LDI) command inserts a line that shows the data item and its value before the first line of the program listing on the Intercept panel. The data item appears on all subsequent Intercept panels until it is reset by the RDI (reset data item) command.

Syntax

Use the following syntax for LDI or KEEP:

LDI data-item|expression



Note: Data for variables are case-sensitive.

You can enter this command in its abbreviated form: K.

Variables

Use the following variables with LDA or AUTOKEEP:

data-item	An elementary data item or group name from the test program.
expression	For assembler programs, enter an expression in register notation.

Usage Notes

Modify elementary items from within the keep window. Type over the value displayed on the screen with the new value.

The LDI command supports group level items. To view the members of a data group, specify the group name as the data-item for the LDI command.

Use the LDI command to view COBOL SPECIAL REGISTERS. For example, this command places the current value of return code on the Intercept panel:

LDI RETURN-CODE

The data-item parameter can be a qualified data item, as in the following examples. Specify either 1 or 2 levels of qualification.

COBOL Use this command to refer to the EMPLOYEE-NO in the MASTER-RECORD if the EMPLOYEE-NO is not unique.

LDI EMPLOYEE-NO IN MASTER-RECORD

PL/I: Use this command to refer to the EMPLOYEE_NO in the MASTER_RECORD if the EMPLOYEE_NO is not unique.

LDI MASTER_RECORD.EMPLOYEE_NO

The data-item parameter can be a fixed or variably subscripted data item or an indexed data item. For example, the following commands show the use of fixed subscripts:

COBOL:

LDI WA-TI-3A(1,2,3)

PL/I:

LDI WA_TI_3A(1,2,3)

You can specify also variable subscripts, as follows:

COBOL:

LDI WA-TI-3A(SUB1,SUB2,SUB3)

PL/I:

LDI WA_TI_3A(SUB1,SUB2,SUB3)

In this case, the value displayed changes as the subscripted variables change. The values of the subscripts are also displayed. An * INVALID * message is placed next to any of the subscripts that are out of range.

The data-item can also be a qualified, fixed or variably subscripted data-item, or an indexed data item, as in the following examples:

COBOL: This command displays the fourth element of the Table WA-1A in WA-TABLE-1:

LDI WA-1A OF WA-TABLE-1 (4)

PL/I: This command displays the fourth element of the Table WA_1A in WA_TABLE_1:

LDI WA_TABLE_1.WA_1A (4)

For PL/I programs, the data-item parameter can be a locator qualified based variable. For example, use the following command to display the allocation of ELEMENT that is based off of NEXT_ELEM_PTR.

LDI NEXT_ELEM_PTR->ELEMENT

Both the data-item and the locator qualifier can also be qualified and fixed or variably subscripted. CA supports up to 15 levels of locator qualification.

COBOL programs can be compiled with the ARITH(EXTEND) parameter which will allow a numeric field to be greater than 18 bytes. These fields will be displayed with the message 'TOO BIG TO DISPLAY' in the value. To view or change the actual value of the field, use the CORE command.

Special Registers

Special registers are compiler-generated storage areas whose primary use is to store information produced through one of COBOL's specific features.

You may display the value of the following special registers in the application:

```
RETURN-CODE
SORT-RETURN
SORT-FILE-SIZE
SORT-CORE-SIZE
SORT-MODE-SIZE
LABEL-RETURN
```

For more information regarding these special registers, refer to your *IBM VS COBOL for OS/VS Manual*, *IBM COBOL II Application Programming Language Reference Manual*, or *COBOL/370 Language Reference Manual*.

LDX (or KEEPX) Command (List Data Item in Hex)

The list data item in hexadecimal (LDX) command inserts three lines showing the data item and its value before the first display lines of the program listing on the Intercept panel. The second and third lines display the data representation in hex format showing zones and numerics, respectively. The data item appears on all following Intercept panels until it is reset by the RDI (reset data item) command.

Syntax

Use the following variables with LDX or KEEPX:

```
LDX data-item
```

You can enter this command in its abbreviated form: KX..

Variables

Use the following variable with the LDX or KEEPX commands:

data-item An elementary data item or group item from the test program.

Usage Notes

Modify elementary items from within the keep window. Type over the value displayed on the screen with the new value. Note that you cannot modify the zones and numerics portion of the data display.

The LDX command supports group level items. To view the members of a data group, enter the group name as the data-item for the LDX command.

Use the LDX command to view COBOL special registers. For example, use the following command:

```
LDX RETURN-CODE
```

The data-item parameter can be a qualified data item, as in the following examples. Specify either 1 or 2 levels of qualification.

COBOL: Use this command to refer to the EMPLOYEE-NO in the MASTER-RECORD if the EMPLOYEE-NO is not unique:

LDX EMPLOYEE-NO IN MASTER-RECORD

PL/I: Use this command to refer to the EMPLOYEE_NO in the MASTER_RECORD if the EMPLOYEE_NO is not unique:

LDX MASTER_REC.EMPLOYEE_NO

The data-item parameter can be a fixed or variably subscripted data item or an indexed data item. For example, the following commands show the use of fixed subscripts:

COBOL:

LDX WA-TI-3A(1,2,3)

PL/I:

LDX WA_TI_3A(1,2,3)

You can specify also variable subscripts, as follows:

COBOL:

LDX WA-TI-3A(SUB1,SUB2,SUB3)

PL/I:

LDX WA_TI_3A(SUB1,SUB2,SUB3)

In this case, the value displayed changes as the subscript variables change. The values of the subscripts are also displayed. An * INVALID * message is placed next to any of the subscripts that are out of range.

The data-item can be a qualified fixed or variably subscripted data item or an indexed data item, as in the following examples:

COBOL: This command displays the fourth element of the table WA-1A in WA-TABLE-1:

LDX WA-1A OF WA-TABLE-1 (4)

PL/I: This command displays the fourth element of the table WA_1A in WA_TABLE_1:

LDX WA_TABLE_1.WA_1A (4)

For PL/I programs, the data-item parameter can be a locator qualified based variable. For example, use the following command to display the allocation of ELEMENT that is based off of NEXT_ELEM_PTR:

LDX NEXT_ELEM_PTR->ELEMENT

Both the data-item and the locator qualifier can also be qualified and fixed or variably subscripted. CA supports up to 15 levels of locator qualification.

Special Registers

See the previous section, LDI (or KEEP) Command (List Data Item), for a description of the supported special registers.

LEQ Command (List Equated Names)

The LEQ command lists the equated symbolic names and data items set by the EQUATE command. Delete the symbolic names by using the DROP command.

Syntax

Use the following syntax with LEQ:

LEQ

LINKAGE Command (Format the COBOL Linkage Section)

The LINKAGE command displays the Linkage Display panel. Modify the data values on this display.

Syntax

Use the following syntax with LINKAGE:

LINKAGE <data-item> <PRINT>

You can enter this command in its abbreviated form: LI.

Variables

Use the following variables with the LINKAGE command:

data-item	Causes a display of the data item, any subordinate data items, and their data values.
PRINT	Causes the data displayed by the command to be written in the session log.

Usage Notes

Consider the following LINKAGE SECTION definition:

```
LINKAGE SECTION
01 PARM-VALUE
   05 PARM-COUNT                PIC S9(4) COMP SYNC.
   05 PARM-DATA
      10 PARM-DATE              PIC X(5).
      10 PARM-TIME.
         15 PARM-HH             PIC XX
         15 PARM-MM             PIC XX
         15 PARM-SS             PIC XX
```

If you enter the LINKAGE command without operands, it formats and displays all of the preceding data items and their data values.

The command LI PARM-DATA formats and displays PARM-DATA, PARM-DATE, PARM-TIME, PARM-HH, PARM-MM, and PARM-SS.

The command LI PARM-SS formats and displays PARM-SS.



Important! The LINKAGE command should not be used until the COBOL program has executed the PROCEDURE DIVISION or ENTRY statement.

The LINKAGE command is not valid for assembler or PL/I programs.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items are from the time before that statement was executed.

LISTAT Command (List Breakpoints)

The LISTAT command displays the Breakpoint Status panel. This is a listing of the unconditional breakpoints set in the currently qualified program. You can delete breakpoints from this panel as well by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing enter.

Syntax

Use the following syntax with LISTAT:

LISTAT

LISTBP Command (List All Breakpoints)

The LISTBP command displays the Breakpoints panel. This is a listing of all conditional (WHEN) and unconditional (AT) breakpoints set in the application. You may also delete breakpoints from this panel by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing Enter.

Syntax

Use the following syntax with LISTBP:

LISTBP

You can enter this command in its abbreviated form: LBP.

LISTLBL Command (List Labels)

The LISTLBL command lists the labels that were added with the .label command.

Syntax

Use the following syntax with LISTLBL:

LISTLBL

LISTUSE Command (List Usings)

The LISTUSE command displays the Using Status panel. This is a listing of all usings set by the USING command. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax with LISTUSE:

LISTUSE

You can enter this command in its abbreviated form: LISTU.

LISTWHEN Command (List WHEN Conditions)

The LISTWHEN command displays the When Conditions Display panel. This is a listing of the conditional breakpoints set in the currently qualified program. It lists both global and local conditional breakpoints. You may delete breakpoints from this panel as well by placing an **X** in the command column of the line the breakpoint to be deleted is listed on, and pressing Enter.

Syntax

Use the following syntax with LISTWHEN:

LISTWHEN

You can enter this command in its abbreviated form: LISTW.

MAP Command (Region MAP Display)

The MAP command invokes the Region Map Display Panel. This panel lets you display information about programs in storage. For more information, see [Core Option \(see page 36\)](#) and [Map Option \(see page 38\)](#).

Syntax

Use the following syntax with MAP:

MAP

MLOG Command (Record Debugging Session)

Logs and monitors a debugging session so you can save it and reuse the session commands in a future debugging session.

Syntax

Use the following syntax with MLOG:

MLOG <START|STOP|CANCEL|DISPLAY|LOAD> <name> <description>

Variables**MLOG START *name description***

Monitoring commands are recorded for later use in restoring the debug session. The monitoring commands are saved into INT1CLIB as a member with the name specified.

Name is optional and is 1-8 characters and must be a valid name for a PDS member. If name is not specified the default name is the name of the loaded program.

Description is optional and is 1-35 characters. This is a comment field where the user can provide a description of the debug session.

After the MLOG START command is invoked, 'RECORDING' will be displayed in the upper right hand corner for the screen. This is an indicator that debugging commands are being recorded for use at a later time.

MLOG STOP

Recording of monitor commands is stopped for the user entering the command.
After the MLOG STOP command is issued the RECORDING indicator will be turned off.

MLOG CANCEL

Logging of monitoring commands is turned off for the session and entries recorded since the prior MLOG START command are deleted.

MLOG DISPLAY

Presents a menu of active and saved debugging sessions in the INT1CLIB file. This is the default if MLOG is specified with no subcommand.

MLOG LOAD *name*

Performs a load of a saved debug session from the INT1CLIB library. This command behaves the same as the INCLUDE command.

NEXT Command (Executing Verbs)

The NEXT command executes the next verb in the program. You will see the Next Intercept panel after entering NEXT.

Syntax

Use the following syntax with NEXT:

NEXT <nnn|P|BEFORE|AFTER|?|OVER|RETURN>

You can enter this command in its abbreviated form: N.

Variables

Use the following variables with the NEXT command:

nnn	Specifies the number of statements to execute. The default is one.
P	Specifies that all statements in the current paragraph that are not to be skipped will be executed, and execution stops at the next paragraph.
BEFORE	Specifies that when the NEXT command is used, execution stops before the statements are executed. Note this is the default.
AFTER	Specifies that when the NEXT command is used, execution stops after the statements are executed.
Notes:	
If NEXT is set to AFTER, statements that cause execution to leave the current program, such as GOBACK for COBOL, will have no effect on the NEXT command.	
If data monitoring was in effect, all data item changes are captured at a point in time <i>before</i> COBOL statements referencing them are executed even when NEXT AFTER is set.	
?	Displays a short message, reporting on whether NEXT stops before or after the specified number of statements.
OVER	

Specifies that when the NEXT command is used executes through a verb, instruction, or statement and return to that line. This is to avoid breakpoints.

RETURN Specifies that when the NEXT command is used to continue execution until the next CALL or PERFORM verb. Execution will stop when it hits a CALL or PERFORM execution.

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.

NOFREQ Command (Remove Frequency Counters)

Use the NOFREQ command to remove the frequency counters and arrows from the Intercept panel.

Syntax

Use the following syntax for NOFREQ:

NOFREQ



Note: This is used along with the FREQ command. This command and the FREQ command have been replaced by the COUNTS command.

OFF (or OFFU) Command (Remove Unconditional Breakpoints)

Use the OFF command to delete unconditional breakpoints set by the AT or UNCOND command, the A or U line command, the) line command, the SKIP command, the S line command, or the Breakpoint panel.

If you specify the OFF command with no operands, the Breakpoint panel is displayed.

Delete a breakpoint by specifying OFF with the statement number where the breakpoint is set in the COMMAND field of the Intercept panel.

Syntax

Use the following syntax for OFF and OFFU:

```
OFF <statement-number|paragraph-name|procedure-name|label-
name|ENTRY|EXIT|LABEL|DBCALL|ALL ENTRY|ALL EXIT|ALL LABEL|ALL DBCALL <(subcommand-
list)>>
```

You can enter this command in its abbreviated form: O.

Variables

Use the following variables with the OFF or OFFU commands:

Statement-number	A program statement number where an existing breakpoint is deleted.
Paragraph-name	A COBOL paragraph name where an existing breakpoint is deleted.
Procedure-name	A PL/I procedure name where an existing breakpoint is deleted.
Label-name	An assembler or PL/I label name where an existing breakpoint is deleted.
ALL	Indicates all unconditional breakpoints be deleted.
ENTRY	Deletes the breakpoint set by the ENTRY operand of the AT or UNCOND command in the currently qualified program.
EXIT	Deletes the breakpoint set by the EXIT operand of the AT or UNCOND command in the currently qualified program.
LABEL	Deletes the breakpoint set by the LABEL operand of the AT or UNCOND command in the currently qualified program.
DBCALL	Deletes the breakpoint set by the DBCALL operand of the AT or UNCOND command in the currently qualified program.
ALL ENTRY	Deletes the breakpoints set by the ALL ENTRY operand of the AT or UNCOND command.
ALL EXIT	Deletes the breakpoints set by the ALL EXIT operand of the AT or UNCOND command.
ALL LABEL	Deletes the breakpoints set by the ALL LABEL operand of the AT or UNCOND command.
ALL DBCALL	Deletes the breakpoints set by the ALL DBCALL operand of the AT or UNCOND command.

Usage Notes

Set a breakpoint by paragraph, procedure, or label name, and delete by the statement number on which the paragraph is declared and vice versa. For example, consider the following code:

```

041400      0000-INITIALIZATION SECTION.
041500
041600      MOVE +20 TO VARIABLE-LENGTH-1.
041700      MOVE +25 TO VARIABLE-LENGTH-2.
041800      MOVE +30 TO VARIABLE-LENGTH-3.

```

Set a breakpoint on the first statement of the example by either of the following UNCOND commands:

```

UNCOND 41400
      or
UNCOND 0000-INITIALIZATION

```

The breakpoint could be removed by either of the following OFF commands:

```

OFF 41400
      or
OFF 0000-INITIALIZATION

```

OFFWN (or OFFC) Command (Remove Conditional Breakpoints)

Use the OFFWN command to delete conditional breakpoints set by the C, W or V line commands, by the WHEN command, or in the When panel.

If you specify the OFFWN command with no operands, the When panel displays.

Delete a breakpoint by specifying OFFWN and when-name of the breakpoint to be deleted in the Command field of the Intercept panel.

Syntax

Use the following syntax for OFFWN and OFFC:

OFFWN <when-name|ALL>

Variables

Use the following variables with the OFFWN or OFFC commands:

when-name A user-defined unique label specified in the WHEN command and used by the OFFWN command. When-name may be from one to eight characters in length. A when-name can also be a statement number. Variable change breakpoints set by the V line command generate their own when-names (see the description of the V line command earlier in this section).

ALL Indicates that all conditional breakpoints should be reset.

POINT Command (Direct PREV and ADVANCE Pointer)

The POINT command causes the PREV and ADVANCE commands to operate from the statement number, paragraph, or procedure, or label name designated in the POINT command.

Syntax

Use the following syntax for POINT:

POINT statement-number|paragraph-name|procedure-name|label-name

Enter this command in its abbreviated form: PO.

Variables

Use the following variables with the POINT command:

statement-number A program statement number from which PREV and ADVANCE are used.

paragraph-name A COBOL paragraph name from which PREV and ADVANCE are used.

procedure-name A PL/I procedure name from which PREV and ADVANCE are used.

label-name An assembler or PL/I label name from which PREV and ADVANCE are used.

Usage Notes

You must turn on the trace facility for the POINT command to function. For example, if the program abended on statement 13100 and you wanted to review the statements that were executed earlier, the POINT command could be used to establish a statement from which to use the PREV and ADVANCE commands. Suppose a part of the trace table contained the following:

```
13100
13000
.
.
.
11000
10900
10800
10700
10300
10200
10100
09900
```

If you entered the statement PO 10300 in the command line, ADVANCE would display and highlight 10700, but PREV would display and highlight 10200.

If data monitoring was in effect when the statement that is highlighted from the POINT command was executed, the values displayed for your data items are from the time before that statement was executed.

PREV Command (Trace in Reverse)

The PREV command causes the display to scroll to the previously executed statement and highlights it. Use the PREV command to step backward through the execution of the program.

Syntax

Use the following syntax for PREV:

```
PREV <nnn|P nn>
```

You can enter this command in its abbreviated form: P.

Variables

Use the following variables with the PREV command:

nnn	Specifies the number of previously executed statements for the application to scroll. The default is one.
P	Specifies that the previously executed paragraph, procedure, or label is displayed.
nn	Specifies the number of previously executed paragraphs, procedures, or labels for the application to scroll. The default is one.

Usage Notes

You must turn on the trace facility for the PREV command to function. You turn on the trace facility by default when you first invoke the application. For further information on turning the trace facility on and off, see the TRACE command later in this section.

The PREV command visually displays the flow of execution of the program in reverse. It does not actually reverse the execution or execute the program in reverse.

The PREV command does not affect the point at which the application resumes execution.

If data monitoring was in effect when the statement that is highlighted from the PREV command was executed, the values displayed for your data items are from the time before that statement was executed.

PS Command (Print Stream)

The Print Stream (PS) command causes the currently displayed stream to be written to the session log for the current session.

Syntax

Use the following syntax for PS:

PS

Usage Notes

The PS command writes the entire stream, not just the portion displayed on the screen. For example, if only the data that is currently shown on the screen display is needed, CA Roscoe users should allocate an INT1PRNT DD data set and use the PRINT PF Key function. Native TSO users can use this command as well. ISPF users should use the ISPF PRINT facility.

QUALIFY Command (Set Current Program Id)

The QUALIFY command identifies the program to which all subsequent commands apply. This qualification is automatically reset by another QUALIFY command, or resulting from the flow of the execution. When testing more than one program, the listing switches automatically to the qualified program.

Syntax

Use the following syntax for QUALIFY:

QUALIFY program-id

Variables

Use the following variables with the QUALIFY command:

program-id Specifies the PROTSYM member name of the program to which the following commands apply.

The program-id of the currently intercepted program is automatically qualified at each program intercept. The qualified program-id of the program is displayed in the upper left corner of the display.

Usage Notes

The QUALIFY command is useful when you are testing multiple programs. For example, you are testing programs A and B. The initial breakpoint occurs in A, but you want to set a breakpoint in B. The following command switches the listing file in the Intercept panel to program B:

```
QUALIFY B
```

Use display commands to find and scroll to the statement where you want to set the breakpoint. Issue the AT command, and the breakpoint is set in program B. Issue the following commands:

```
QUALIFY A
GO
```

Your test resumes execution until the breakpoint in program B is encountered.

QUIT Command (Terminate Session)

The QUIT command stops the test session. The Execution Control panel displays so you can do additional testing. Pressing the END key twice in a row is the same as entering the QUIT command.

Syntax

Use the following syntax for QUIT:

```
QUIT
```

Usage Notes

The QUIT command is the same as END and CANCEL.

The first time you press the END key, the message USE "END" TO TERMINATE appears at the top of the screen. Press the END key again.

RDI or REMOVE Command (Reset Data Item)

The RDI command removes from the Intercept panel the lines placed there by the LDI (list data item) or the LDX (list data item in hex) commands, or the L or H line commands, and moves the program listing up.

Syntax

Use the following syntax for RDI or REMOVE:

```
RDI data-item|ALL
```

You can enter this command in its abbreviated form: RE.

Variables

Use the following variables with the RDI or REMOVE command:

data-item	Must be a data item in a LDI command.
ALL	Specifies that all LDI items should be deleted from the Intercept panel.

RECORDS Command (Show COBOL Record Formats)

The RECORDS command displays the COBOL File Section Display panel. The File Section Display panel shows the record formats under the file descriptions (FDs). You can modify data values on this display.

Syntax

Use the following syntax for RECORDS:

```
RECORDS <data-item> <PRINT>
```

You can enter this command in its abbreviated form: REC.

Variables

Use the following variables with the RECORDS command:

data-item	Causes a display of the data item, any subordinate level data items, and their data values.
PRINT	Causes the data displayed by the command to be written in the session log.

Usage Notes

Consider the following COBOL RECORD definition:

```
FILE SECTION.
FD INPUT-FILE.
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS F.
01 INPUT-RECORD.
    05 INPUT-NUMBER                PIC X(4) .
    05 INPUT-NAME                  PIC X(36) .
    05 INPUT-SSN.
        10 INPUT-SSN-XXX          PIC X(3) .
        10 INPUT-SSN-YY           PIC X(2) .
        10 INPUT-SSN-ZZZZ        PIC X(4) .
```

If you enter the RECORDS command without operands, it formats and displays all of the previous data items and their values.

The command REC INPUT-SSN formats and displays INPUT-SSN, INPUT-SSN-XXX, INPUT-SSN-YY and INPUT-SSN-ZZZZ.

The command REC INPUT-SSN-ZZZZ formats and displays INPUT-SSN-ZZZZ.

The INPUT-FILE must be open to modify data with the RECORDS command.

The RECORDS command is not valid for assembler or PL/I.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the values displayed for your data items will be from the time before that statement was executed.

REFRESH Command

Use the REFRESH subcommand when you have split the screen during test execution, and have used one screen to change the value of a data item that is currently displayed on the other screen (screen 2).

REFRESH, typed on screen 2, lets the application pick up and display the newly updated value.

Syntax

Use the following syntax for REFRESH:

REFRESH

REGS Command

The REGS command displays the current application's registers. This command is only valid for assembler programs.

Syntax

Use the following syntax for REGS:

REGS <ON|OFF|DISPLAY|FLOAT|VECTOR>

You can enter this command in its abbreviated form: REG.

Variables

Use the following variables with the REGS command:

ON	Displays the general registers in the Keep window.
OFF	Removes the general registers from the Keep window.
DISPLAY	Displays a panel where the general registers, their high halves, and the access registers are displayed.
FLOAT	Displays a panel where the floating point registers are displayed.
VECTOR	Displays a panel where vector registers are displayed.

RESET Command

RESET removes column numbers that are displayed on the Intercept or Display panel by the COLUMN command.

Syntax

Use the following syntax for RESET:

RESET

Usage Notes

The RESET command runs from any panel that has a command line that accepts the COLUMN command.

RESTART Command

The RESTART command re-initiates a batch link job to the beginning of the current step being debugged.

RESTART

Usage Notes

This command is only valid under batch link. Foreground users should use the QUIT command.

RUN Command (Resume Execution)

The RUN command resumes execution of the program ignoring all breakpoints. The program continues to execute until normal termination or until it abends. The Execution Control panel displays.

This command removes any capability for further program debugging. The application will *not* intercept abends.

However, if an optional statement number is passed to the RUN command, when execution from the RUN command hits that statement, the RUN command is converted into a GO command. This means that existing breakpoints are honored again.



Note: Using the RUN command during a batch link session also issues a SUSPEND command, releasing the terminal.

Syntax

Use the following syntax for RUN:

RUN <statement-number>

Variables

Use the following variables with the RUN command:

Statement-number	A program statement number where the RUN command is converted to a GO command.
-------------------------	--

SDWA Command (System Diagnostic Work Area)

The SDWA command displays the System Diagnostic Work Area if a failure occurs. The panel displayed gives the following information: the SDWA address, the ABEND code, the PSW, the program name, the EP address, the offset, and the contents of the registers at the time of failure.

Syntax

Use the following syntax for SDWA:

SDWA

SET Command (Change Data Item Value)

Use the SET command to change the value of a data item.

Syntax

Use the following syntax for SET

SET data-item <=> value|LOW VALUES|HIGH VALUES|SPACES|ZERO|ZEROES|X'hex char string'



Note: Data for variables are case-sensitive.

Use the following variables with the SET command:

data-item	A data item in the monitored program or a COBOL special register. Data-item may be a fixed or variably subscripted data item or an indexed data item. It may also be a qualified data item.
value	The new value of the data item.
LOW-VALUES	Indicates that the value of the data item is the figurative COBOL constant of hexadecimal '00', which is the lowest value in the computer's collating sequence.
HIGH-VALUES	Indicates that the value of the data item is the figurative COBOL constant of hexadecimal 'FF', which is the highest value in the computer's collating sequence.
SPACES	Indicates that the value of the data item is one or more blanks or spaces.
ZEROS	Indicates that the value of the data item is one or more zeros.
ZEROES	
hex-char-string	1 to 20 valid hexadecimal characters. The data is replaced from left to right.

Usage Notes

Alter data by overtyping on any of the data displays (from DISPLAY, LINKAGE, and RECORDS command). Change elementary data items using the SET command on the Intercept panel.

SET EMPLOYEE-ID = '123-45-6789'

This entry sets the data item EMPLOYEE-ID to a value of 123-45-6789. Use this command with the command library facility to initialize areas of data for testing. [See the INCLUDE command described earlier in this section for further information on the command library facility.](#)

Modify fixed or variably subscripted data items or index data items, that is, table entries, using the SET Command, as in the following examples:

COBOL: SET WA-TI(4) = 6

Sets the fourth element in the table WA-T1 to 6.

SET WA-TI2(SUB1,SUB2) = 'S1S2'

Sets an element in WA-T2 depending on the values of SUB1 and SUB2 to S1S2.

SET WA-T1-1A OF WA-TABLE-1 (2)= 2

Sets the second element of the non-unique table WA-T1-1A in the unique table WA-TABLE-1 to 2.

PL/I: SET WA_TI(4) = 6

Sets the fourth element in the table WA_T1 to 6.

SET WA_TI2(SUB1,SUB2) = 'S1S2'

Sets an element in WA_T2 depending on the values of SUB1 and SUB2 to S1S2.

SET WA_TABLE_1.WA_T1_1A(2) = 2

Sets the second element of the non-unique table WA_T1_1A in the unique table WA_TABLE_1 to 2.

SET WA_TBL_PTR->WA_T1_1A(2) = 2

Sets the second element of the based table WA_T1_1A in the table allocation pointed to by WA_TBL_PTR.

If you are navigating the trace table and data monitoring was in effect for the currently highlighted statement, the SET command is not enabled. You must first issue the CS command to reset the tracing and then issue the SET command.

COBOL programs can be compiled with the ARITH(EXTEND) parameter which will allow a numeric field to be greater than 18 bytes. The SET command is unavailable to modify fields longer than 18 bytes. The CORE command can be used to edit these fields.

SKIP Command (Skip a Statement)

The SKIP command causes the statement number specified to be skipped when the program is executed.

The Intercept panel shows an S to the left of any statements that are to be skipped.

Skip is considered an unconditional breakpoint and may be reset using the OFF control command or the O or X line command.

Syntax

Use the following syntax for SKIP:

SKIP statement-number

Variables

Use the following variable with the SKIP command:

statement-number	Specifies the statement-number in the program that is to be skipped when the program is executing.
-------------------------	--

Usage Notes

Skipping a statement causes the execution of that statement to be bypassed entirely, with execution resuming at the verb or label whose statement number is nearest to and greater than the statement being skipped. In some instances, this may not result in the required execution path, with execution resuming at a statement that would otherwise not have been executed.

This can occur when the statement being skipped is any of the following statements:

- The last statement in the program
- The last statement in a PERFORMed paragraph
- Immediately followed by an ELSE statement



Important! In those instances that may not result in the required execution path, you should avoid using a SKIP command, and instead use an unconditional breakpoint with an associated GO command.

To set an unconditional breakpoint, type AT on the command line and press Enter. When prompted, enter the statement number where the breakpoint should occur, which is the statement you wish to skip. Type **GO nnnnn** in the Command field, where *nnnnn* is the number of the statement at which execution should resume.

In this example, a SKIP command on statement 917 would cause execution to resume at statement 919, which is not the required execution path:

```

000915          IF A NOT EQUAL ZERO
000916          MOVE A TO OPTION-CHOSEN
S 000917          PERFORM PROCESS-SELECTION
000918          ELSE
000919          DISPLAY 'ERROR: INVALID SELECTION'
000920          MOVE 12 TO RETURN-CODE.
000921          PERFORM LOG-RESULT.
```

Instead, use the AT command to force execution to resume at statement 921:

```

----- CA InterTest Batch BREAKPOINT PANEL -----
OPTION ==> S

S - SET A BREAKPOINT
R - RESET A BREAKPOINT
L - OR BLANK, LIST THE BREAKPOINTS

SPECIFY PROGRAM NAME AND STATEMENT NUMBER:
```



```

PROGRAM NAME ===>
STATEMENT    ===> 917

COUNT      ===>          AFTER ===> N  (Y/N)

COMMANDS TO BE EXECUTED AT BREAKPOINT:
===> GO 921

```

You should also be aware that the use of a SKIP command may result in incorrect results when the program has been optimized, either by the IBM optimize option or by CA Optimizer or CA Optimizer /II. Skipping a statement in an optimized program may result in the loss of calculations or intermediate results, which may affect future statements.

SLOW (or AUTOSTEP) Command (Slowly Resume Program Execution)

The SLOW command resumes execution of the program one verb at a time and displays the Intercept panel between the executions of each verb.

Syntax

Use the following syntax for SLOW or AUTOSTEP:

```
SLOW      <nnn|FAST>
```

You can enter this command in its abbreviated form: AUTO.

Variables

Use the following variables with the SLOW command:

nnn Specifies the number of seconds to halt the display between verb executions. The default is two seconds.

FAST Executes the program without halting between screen refreshes.

Usage Notes

AUTOSTEP (or AUTO) is a synonym for SLOW.

Interrupt this feature by pressing the Attention key. For information on the Attention key, see [Stop a Looping Program in Guidelines for Novice Users \(see page 119\)](#).

Occasionally when you press the Attention key, the program stops and the short message field SUBCOMMAND ABEND is displayed. If this occurs, issue the NEXT command to refresh the display.

SNAP Command (Produce a SNAP Report)

The SNAP command produces a CA SymDump Batch, CA Optimizer/II, or CA Optimizer SNAP report, if you have one of these products initialized. For details on the SNAP report, see the [appropriate product's documentation](#).

Syntax

Use the following syntax for SNAP:

```
SNAP
```



Note: Minimum release requirements are CA SymDump Batch 2.0, CA Optimizer/II 3.0, or CA Optimizer 7.0.

SPEED Command (Accelerate Program Testing)

The SPEED command deactivates the trace and frequency facilities.

Syntax

Use the following syntax for SPEED:

SPEED <ON|OFF>

Variables

Use the following variables with the SPEED command:

ON Turns off trace and frequency counting. Issues the TRACE OFF and the FREQ OFF commands.

OFF Enables tracing and statement frequency counting. Issues the TRACE ON and the FREQ ON commands.

Usage Notes

This feature is very useful when testing a large program. It turns off trace and frequency counting and therefore improves performance. For example, to test a COBOL program with 20,000 PROCEDURE DIVISION statements, set a breakpoint at statement 15000, and step through the program when you get to that point. The commands to perform this are as follows:

```
AT 15000
SPEED
```

When statement 15000 is reached, an Intercept panel is displayed. To step through the program from this point, enter the following to turn on trace and frequency counting:

```
SPEED OFF
STEP 1
```

STEP Command (Set Count for GO Command)

The STEP command sets the execution mode to step operation. For each GO command, the specified number of verbs is executed, and the STEP COUNT Intercept panel is displayed when that number of verbs has executed.

Syntax

Use the following syntax for STEP:

STEP <nnnn|BEFORE|AFTER>

You can enter this command in its abbreviated form: ST.

Variables

Use the following variable with the STEP command:

nnnn	Specifies the number of verbs to be executed each time you enter a GO command. Setting the STEP count to 0 (zero) turns off step counting so that the next GO command executes until a breakpoint is encountered, an abend occurs, or the program reaches normal termination. The maximum step count that you can enter is 9999.
BEFORE	Specifies that execution should stop before the statement that satisfies the STEP count. This is the default.
AFTER	Specifies that execution should stop after the statement that satisfies the STEP count.

Notes:

If STEP is set to AFTER, statements that cause execution to leave the current program, such as GOBACK for COBOL, have no effect on the STEP count.

If data monitoring was in effect, all data item changes are captured at a point in time *before* COBOL statements referencing them are executed even when STEP AFTER is set.



Note: If no variable is used with the STEP command, the current STEP count and status is displayed in the upper right hand corner.

SUSPEND Command (Suspend a Batch Link Session)

Use the SUSPEND command to detach a terminal from a batch link debugging session without terminating the session.

Syntax

Use the following syntax for SUSPEND:

SUSPEND

You can enter this command in its abbreviated form: SUS.

Usage Notes

The SUSPEND command is only valid for batch link debugging sessions and you can only use it from the Monitor Control or Intercept panels.

A suspended batch link session becomes immediately available for selection from the Batch Link Selection panel.

Reconnect a suspended session by the same user or by a different user. Regardless of which user selects the suspended session, the session is resumed from exactly the same point at which the session was suspended, and continues to use the same profile member that was defined in the INT1OPTS file.

TRACE SOURCE Command (Trace with Source Code Displayed)

The TRACE SOURCE command functions similar to the TRACE command, but displays the code being traced as well as the statement number.

Syntax

Use the following syntax for TRACE SOURCE:

```
TRACE SOURCE n
```

You can enter this command in its abbreviated form: TR SO.

Variables

Use the following variables with the TRACE SOURCE command:

n	Specifies the maximum number of trace entries to be displayed. The default is 500.
----------	--

TRACE Command (Controls Program Trace Entries)

If the TRACE command does not have an operand, it displays the Program Trace Display panel. This panel shows statement numbers in the order they were executed.

Syntax

Use the following syntax for TRACE:

```
TRACE <ON|OFF|n>
```

You can enter this command in its abbreviated form: TR.

Variables

Use the following variables with the TRACE command:

ON	Turns on statement tracing. The number of entries retained is set by the TRACE(n) parameter in the initialization PARMLIB member. The default is 500.
-----------	---

OFF	Turns off statement tracing.
------------	------------------------------

n	Specifies the number of entries to be saved in the trace table and displayed by the TRACE command. The current trace table is cleared and reset. This overrides the value specified in the TRACE(n) initialization PARMLIB member.
----------	--

Usage Notes

The trace setting is remembered from session to session.

It is possible to view the statements that have been executed in reverse by using the PREVIOUS command, instead of having to view the trace display.

If data monitoring is in effect for any program in your application, changing the trace table resets the effects of the DATAMON command.

TRP Command (Trace Paragraphs)

The TRP command invokes the Program Trace Display panel. This panel shows statement numbers and paragraph, procedure, or label names executed.

Syntax

Use the following syntax for TRP:

TRP

Usage Notes

The trace facility must be enabled for the TRP command to function.

USING Command (Assign a Register or Address to a DSECT or CSECT)

The USING command lets you assign a register or address for the debugger to use to map a DSECT or CSECT that has no current using in the program being debugged. This command is only valid when debugging an assembler program.

Syntax

Use the following syntax for USING:

USING dsect|csect register|address

Variables

Use the following variables with the USING command:

dsect	Assembler DSECT that you wish to map
csect	Assembler CSECT that you wish to map
register	Register that maps the DSECT or CSECT
address	Address that maps the DSECT or CSECT

WHEN (or COND) Command (Set Conditional Breakpoints)

The WHEN command sets conditional breakpoints. There are two types of conditional breakpoints and the format of the command syntax is different for each:

Use this Syntax format	For this Type of Conditional Breakpoint
1	Whenever the value of a specified data item changes
2	When a relationship between two data items is true, for example, 'SUBSCRIPT-1 GT 4'

Enter operands in the When panel or in the COMMAND field of the Intercept panel in one of the following formats:

Format 1 This type stops the program's execution any time the value of the specified data item changes. For example: WHEN whenname1 R1498_STATUS.

Format 2 This type checks a relational expression for a TRUE condition before intercepting the program. For example:

WHEN whenname2 R1498_STATUS GT G

If the *when-name* specified is a statement number, the WHEN condition is only tested before executing that statement number. In this case, the application does not test for the condition before each statement. This local type of conditional breakpoint reduces computing overhead.



Notes:

- WHEN conditions that are set for subscripted or indexed items are checked only if the subscript or index is valid.
- If you use the WHEN command without operands, the WHEN panel is displayed. Set, reset, or list conditional breakpoints from this displayed WHEN panel.

Syntax Format 1 When the value of a specified data Item changes

Syntax

Use the following syntax for WHEN or COND:

WHEN <when-name data-item-1> <BEFORE|AFTER> <(subcommand-list)>

You can enter this command in one of its abbreviated forms: WH, WN or C.

Variables

Use the following variables with the WHEN or COND commands:

when-name	A user-defined unique label or a statement number; if the when-name is a statement number, the when condition is only checked before the statement is executed. It is also displayed in the title line of the W when-name Intercept panel.
data-item-1	The data item is examined before each verb is executed unless the when-name is a statement number in which case the data item is only examined before executing that statement (local conditional breakpoint). Whenever the value of the data item changes, a W when-name Intercept panel is displayed. Data-item-1 may be a fixed or variably subscripted, indexed, or qualified data item.
BEFORE	Specifies that execution should stop <i>before</i> the statement when the condition is met. (That is, the application detected that the value of the specified <i>data-item</i> has changed.) This is the default behavior.
AFTER	Specifies that execution should stop <i>after</i> the statement when the condition is met. (That is, the application detected that the value of the specified <i>data-item</i> has changed.)

subcommand-list A list of application commands, separated by semicolons, to be executed when the breakpoint is encountered during the test. The subcommand-list is enclosed in parenthesis.

Syntax

Use the following syntax for WHEN or COND:

```
WHEN <when-name <data-item-1 <operator <data-item-2|value> <BEFORE}AFTER>>
      <(subcommand-list)>>>
```

You can enter this command in one of its abbreviated forms: WH, WN or C

Variables

Use the following variables with the WHEN or COND commands:

data-item-1	Based on the value of the operator, data-item-1 is compared to data-item-2. Whenever the expression is true, a W when-name Intercept panel is displayed. Data-item-1 may be a fixed or variably subscripted, indexed, or qualified data item.
operator	A relational operator used to compare data-item-1 to data-item-2. Values can be any one of the following: EQ or = Equal to NE or != Not equal to LE or <= Less than or equal to GE or >= Greater than or equal to LT or < Less than GT or > Greater than Note: => and =< are not permitted.
data-item-2	Compared to data-item-1. When the comparison is true, a W when-name Intercept panel is displayed. A value can be used in place of data-item-2. Data-item-2 may be fixed subscripted, an indexed data item, or a qualified data item.
value	Specifies a constant value, which you can use in place of data-item-2. This value may be one of the following figurative COBOL constants: LOW-VALUES, HIGH-VALUES, ZEROS, or SPACES. This value may also be hex data in the format following: X'hex-char-string' where hex-char-string is 1 to 20 valid hexadecimal characters.
BEFORE	Specifies that execution should stop <i>before</i> the statement when the condition is met. (The specified <i>condition</i> is TRUE.) This is the default behavior.
AFTER	Specifies that execution should stop <i>after</i> the statement when the condition is met. (The specified <i>condition</i> is TRUE.)

subcommand- A list of application commands, separated by semicolons, to be executed when the **list** breakpoint is encountered during the test. The subcommand-list is enclosed in parenthesis.

Usage Notes

Global conditional breakpoints (the when-name is not specified using a statement number) are breakpoints whose conditions are checked before every executed statement and are very resource intensive and may require considerable overhead. When possible, use local conditional breakpoints (the when-name is specified using a statement number) which are only checked at the time a particular statement is executed. See the Line Commands section earlier in this article for information on setting local conditional breakpoints with the line commands C, W, and V.

A *before* breakpoint is a breakpoint that stops *before* a statement is executed. A *before* breakpoint will never stop before a statement that *makes* the condition true. It would only stop before the next statement which follows the statement that makes the condition true. Thus, a *before* WHEN condition is evaluated before a statement is executed, and if true, the breakpoint occurs immediately, before that statement is executed.

An *after* breakpoint is a breakpoint that stops *after* a statement that makes the condition true. Thus, an *after* WHEN condition is evaluated after the statement is executed, and if true, the breakpoint occurs immediately, after the statement is executed.

* Command (Add Comments in Session Log)

The * command lets you specify a comment to be inserted in the session log. The text following the * is inserted in the current session log exactly as typed.

Syntax

Use the following syntax for *:

```
* text
```

Variables

Use the following variable with the /clist command:

Text Comment to be inserted in the session log

/clist Command (Execute CAMRCMD)

This product provides a facility for executing a CLIST from within the application. By typing in a slash (/) followed by a string, the CAMRCMD CLIST is called with a parameter, which is the string that follows the slash. Review CLIST CAMRCMD for further modification instructions.

Syntax

Use the following syntax for /clist:

```
/clist
```

Variables

Use the following variable with the /clist command:

Clis The parameter with which the CAMRCMD CLIST is called.

Report Commands

The software provides report commands to create reports that you can print and analyze offline. The reports show path coverage of program execution and a histogram report displaying the execution frequencies by statement number in graphic format. Both reports have an execution summary showing the number of executable statements, the number of statements executed, and the percentage of path coverage.

- [HISTOGRAM Command \(Graph Execution Frequencies\) \(see page 113\)](#)
- [XSUM Command \(Execution Summary Report\) \(see page 115\)](#)
- [Session Log Facility \(Review Debugging Session\) \(see page 117\)](#)



Note: The frequency counter must be on (using the COUNTS command) during testing if any of these reports are to be produced.

The reports use the ddname INT1REPT. The allocation of INT1REPT can be either to a SYSOUT class or to a disk data set. Sample allocations are as follows.

Allocation of INT1REPT to a SYSOUT class:

```
ALLOC DD(INT1REPT) SYSOUT(A)
```

Allocation of INT1REPT to a disk data set:

```
ALLOC DD(INT1REPT) DSN(INT.REPORT) NEW SPACE(2 1) CYL
```

If the ddname is allocated to a disk data set, the issuance of each report command completely rewrites the data set. If you would like to obtain both reports, perform the following steps:

1. Allocate the ddname INT1REPT to a data set.
2. Enter the HIST command.
3. Reallocate the INT1REPT to another data set.
4. Enter the XSUM command.

HISTOGRAM Command (Graph Execution Frequencies)

The HISTOGRAM command writes an execution histogram to ddname INT1REPT. The histogram consists of one line per executable statement. The line contains the statement number, the execution frequency counter, and a graphic representation of the number of times the statement has been executed.

The final page of the report identifies the number of statements in the program, an indication of the number of statements that were executed, and the percentage of the statements that were executed during this test.

Syntax

Use the following syntax for HISTOGRAM:

HIST <scale>

You can enter this command in its abbreviated form: HI.

Use the scale parameter to specify the histogram scale to be used. For example, a scale value of 10 would cause one asterisk to be printed on the histogram for every 10 executions of a statement.

If you do not specify a scale, the application automatically selects a scale in a multiple of five that causes no overflow to occur.

The histogram produces correct output only if the test was produced with **FREQ ON** and **TRACE ON**. A sample histogram report follows:

```
1CA InterTest/Batch      Execution Histogram For Program-ID: CAMRC0B2      Date
0
000875 0001 |*
000876 0001 |*
000886 0001 |*
000887 0001 |*
000889 0001 |*
000890 0001 |*
000891 0001 |*
000895 0002 |**
000896 0002 |**
000897 ---- |
000899 0002 |**
000900 0002 |**
000901 0001 |*
000902 0001 |*
000904 ---- |
000906 0001 |*
000908 ---- |
000909 0002 |**
000910 0002 |**
000914 0001 |*
000915 0001 |*
000917 0001 |*
000920 0001 |*
000921 ---- |
000923 ---- |
000925 0001 |*
000931 0001 |*
000932 0001 |*
000933 ---- |
000934 ---- |
000935 ---- |
000938 ---- |
000945 ---- |
000946 ---- |
000953 ---- |
000954 ---- |
000957 ---- |
000958 ---- |
```

A sample histogram report summary follows:

1CA InterTest/Batch Execution Histogram For Program-ID: CAMRC0B2 Date
 -Session Number: 1,890 For Userid user99
 -Number of executable statements: 236
 0Number of statements executed during this test: 78
 0Percentage of statements executed during this test: 33

XSUM Command (Execution Summary Report)

The XSUM command writes an Execution Summary report to ddname INT1REPT. The summary report consists of the program listing with the execution frequency counter to the left of each valid executable statement.

The final page of the report identifies the number of executable statements in the program, an indication of the number of statements that were executed, and the percentage of the statements that were executed during this test.

Syntax

Use the following syntax for XSUM:

SUM <UNEXEC>

Variables

Use the following variable with the XSUM command:

- **UNEXEC**

Specifies that the report should only contain statements that have not been executed.

A sample XSUM report follows:

```
PP 5740-CB1 RELEASE 2.3   JULY 24, 2014                      IBM OS/V5 COBOL
16.20.10  DATE FEB    3,2014

          1                      16.20.10                      FEB 3,2014

          100010  IDENTIFICATION DIVISION.
          100020  PROGRAM-ID.  SCRIPT.
          100030  DATE-COMPILED.  FEB 3,2014.
          100050  ENVIRONMENT DIVISION.
          100060  DATA DIVISION.
          100070  WORKING-STORAGE SECTION.
          100080      01  BIN-REC.
          100090          05  BIN1          PIC 9999 VALUE 0.
          100100          05  BIN2          PIC 9999 VALUE 0.
          100110          05  BIN3          PIC 9999 VALUE 0.
          100120          05  BIN4          PIC 9999 VALUE 0.
          100130          05  MOVE-DISP     PIC X(6).
          100140*
          100150  PROCEDURE DIVISION.
          100160*
0001      100170  INITIALIZATION SECTION.
0001      100180      DISPLAY 'PROGRAM STARTING'.
0001      100190  BIN SECTION.
0010      100200  LOOP-MOVE.
0010      100210      IF BIN1 > 8
          100220          OR BIN2 > 346
          100230          OR BIN3 > 77
          100240          OR BIN1 NOT = BIN4
          100250          OR MOVE-DISP = 'INTCOB'
0001      100260          MOVE ZERO TO BIN1
0001      100270          GO TO LOOP-MOVE-END.
0009      100280          ADD 1 TO BIN1.
```

```

0009      100290      ADD 3 TO BIN2.
0009      100300      ADD 9 TO BIN3.
0009      100310      ADD 1 TO BIN4.
0009      100320      MOVE BIN1 TO MOVE-DISP.
0009      100330      GO TO LOOP-MOVE.
0001      100340      LOOP-MOVE-END.
0001      100350      DISPLAY MOVE-DISP.
0001      100360      MOVE ZERO TO BIN3.
0022      100370      NEXT-LOOP.
0022      100380      IF BIN3 > 20
0001      100390          GO TO NEXT-LOOP-END.
0021      100400      ADD 1 TO BIN3.
0021      100410      GO TO NEXT-LOOP.
0001      100420      NEXT-LOOP-END.
0001      100430      DISPLAY MOVE-DISP.
0001      100440      MOVE ZERO TO BIN2.
0001      100450      LAST-LOOP.
0041      100460      IF BIN2 < 40
0040      100470          ADD 1 TO BIN2
0040      100480          GO TO LAST-LOOP.
0001      100490      LAST-LOOP-END.
0001      100500      DISPLAY BIN2.
0001      100510      DISPLAY 'END OF PROGRAM'.
0001      100520      STOP RUN.

```

A sample XSUM report summary follows:

```

CA InterTest Batch      EXECUTION SUMMARY REPORT FOR PROGRAM-ID: SCRIPT

DATE: 02/03/14 TIME: 16:21:07

SESSION NUMBER: 0127 FOR USERID USER01

NUMBER OF EXECUTABLE STATEMENTS: 000032

NUMBER OF STATEMENTS EXECUTED DURING THIS TEST: 000032

PERCENTAGE OF STATEMENTS EXECUTED DURING THIS TEST: 100

```

A sample XSUM UNEXEC report follows:

```

UNEXECUTED STATEMENTS AND PARAGRAPHS REPORT
----> 000457      MOVE 'B' TO R1498-STATUS
----> 000464      MOVE 'T' TO R1498-STATUS
----> 000468      MOVE 'H' TO R1498-STATUS.
----> 000478      PERFORM 900-WRITE-ERROR THRU
----> 000480      GO TO RETURN-TO-OPSYS.
----> 000497      GOBACK.
----> 000513      GO TO DEMONSTRATE-SLOW-COMMAND.
----> 000515      GO TO DEMONSTRATE-CONDITIONAL-BKPT.
----> 000517      GO TO DEMONSTRATE-SPLIT-SCREEN.
----> 000519      GO TO DEMONSTRATE-LOOP-DETECTION.
----> 000521      GO TO DEMONSTRATE-PREV-COMMAND.
----> 000523      GO TO DEMONSTRATE-PROCESS-TABLE.
----> 000525      GO TO DEMONSTRATE-HIST-COMMAND.
----> 000531      DEMONSTRATE-SLOW-COMMAND.
----> 000532      MOVE ZERO TO OPTION-CHOSEN.
----> 000537      SLOW-COND.
----> 000538      MOVE SPACES TO X-AXIS.
----> 000539      MOVE ZEROES TO YTD, TOTAL-GROSS.
----> 000541      PAY-CALC.
----> 000542      PERFORM POPULATE-GRAPH THRU POPULATE-GRAPH-EX
----> 000545      PAY-CALC-EX.
----> 000546      MOVE SPACES TO CHEQUE-LINE.
----> 000547      MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
----> 000548      PAY-RETURN.
----> 000549      GO TO OPTIONS.
----> 000551      POPULATE-GRAPH.
----> 000552      MOVE MONTH-ITEM(SUB-4) TO X-AXIS-YTD.
----> 000553      MOVE TOKEN-ITEM TO X-AXIS-R(SUB-4).

```

```

----> 000554          ADD  MONTHLY-AMOUNT(SUB-4) TO YTD.
----> 000555          MOVE YTD                      TO TOTAL-GROSS.
----> 000556          POPULATE-GRAPH-EX.
----> 000557          EXIT.
----> 000558          DEMONSTRATE-CONDITIONAL-BKPT.
----> 000559          MOVE ZERO      TO OPTION-CHOSEN.
----> 000561          CONDITIONAL-BREAKPOINT.
----> 000562          MOVE ZEROES TO BILL-YTD, SUB-TOTAL.
----> 000568          PERFORM BILL-CALC THRU BILL-CALC-EX
----> 000572          MOVE SPACES      TO CHEQUE-LINE.
----> 000573          MOVE TOTAL-GROSS TO CHEQUE-AMOUNT.
----> 000574          GO TO OPTIONS.
----> 000576          BILL-CALC.
----> 000577          ADD BILLING-AMOUNT(SUB-7) TO SUB-TOTAL.
----> 000578          MOVE SUB-TOTAL          TO BILL-YTD.
----> 000579          BILL-CALC-EX.
----> 000580          EXIT.
----> 000583          DEMONSTRATE-SPLIT-SCREEN.
----> 000584          MOVE ZERO      TO OPTION-CHOSEN.
----> 000589          SPLIT-SCREEN.
----> 000590          MOVE R1498-TIME0 TO R1498-TIME.
----> 000591          GO TO OPTIONS.
----> 000600          DEMONSTRATE-PREV-COMMAND.
----> 000601          MOVE ZERO      TO OPTION-CHOSEN.

```

Session Log Facility (Review Debugging Session)

The application session log facility records and subsequently enables a user to review the activities of a debugging session.

The session log contains the following:

- An entry identifying the user ID and session number
- Two lines for each program intercept. The first identifies the program and intercept reason code. The second is a copy of the TRACE line as it appeared on the Intercept panel.
- All input commands
- Any comments entered by the user using the * command on the Intercept panel during the testing session
- Any WORKING STORAGE SECTION, LINKAGE SECTION, or RECORDS data displayed using the DI, LI, or RE command with the PRINT option or the P line command

A sample session log with dynamic symbolic support deactivated follows:

```

*INTERTEST BATCH SESSION #1168 FOR USER USER01  BEGINS AT 12:59:30 ON 09/09/14.
*SYMBOLIC FOR CAMRASM LOADED FROM==> NDVRTS.USER01.PROTSYM
* INTERCEPT IN PROGRAM CAMRASM  AT #000005 REASON: *INITIAL*
*
AT START
AT GOBACK
GO

* INTERCEPT IN PROGRAM CAMRASM  AT #000025 REASON: UNCOND BEFORE
*000025
GO

* INTERCEPT IN PROGRAM CAMRASM  AT #000147 REASON: ABEND S0C7
*000147 000145 000144 000142 000141 000140 000139 000137 000136 000135 000134
SET TASKNUM = 0
GO

```

```
* INTERCEPT IN PROGRAM CAMRASM AT #000265 REASON: UNCOND BEFORE
*000265 000260 000259 000258 000257 000255 000254 000253 000252 000251 000250
XSUM
GO
```

A sample session log with dynamic symbolic support activated follows:

```
*INTERTEST BATCH SESSION #1169 FOR USER USER01 BEGINS AT 13:10:30 ON 03/29/14.
* SYMBOLIC FOR: CAMRASM AUTO POPULATED TO: NDVRTS.USER01.PROTSYM
* SYMBOLIC FOR: CAMRASM LOADED FROM=> NDVRTS.USER01.PROTSYM
* INTERCEPT IN PROGRAM CAMRASM AT #000005 REASON: *INITIAL*
*
AT START
AT GOBACK
GO

* INTERCEPT IN PROGRAM CAMRASM AT #000025 REASON: UNCOND BEFORE
*000025
GO

* INTERCEPT IN PROGRAM CAMRASM AT #000147 REASON: ABEND S0C7
*000147 000145 000144 000142 000141 000140 000139 000137 000136 000135 000134
SET TASKNUM = 0
GO

* INTERCEPT IN PROGRAM CAMRASM AT #000265 REASON: UNCOND BEFORE
*000265 000260 000259 000258 000257 000255 000254 000253 000252 000251 000250
XSUM
GO
```

The application session log facility records and subsequently enables a user to review the activities of a debugging session as shown in the previous panel. The session log may be copied into a member of a partitioned data set and subsequently edited and used as the subject of an INCLUDE command.

Pre-allocating the ddname INT1CLOG activates the session log. Allocate the log to a disk file or to a SYSOUT class. In either case, there is no need to specify any DCB information as the application sets it to:

```
RECFM=FB,LRECL=80,BLKSIZE=6160
```

Because user ID keeps the session log, each user who wants to use the log facility must have a session log data set. The data set must be a sequential file. A sample CLIST, INT1CLOG, exists in the CAI. CAVHCLS0 data set to create a sample session log data set.

Testing Procedures

This section describes how to do the following procedures:

- Prepare a program for testing.
- Identify to the application the program to be tested.
- Start and stop program testing at selected statements.
- Test IMS applications.
- Test DB2 applications.

Guidelines for Novice Users

This section provides an overview and checklist for novice users of the Foreground option. The following sections assist you in preparing your programs for testing and using the application panels and commands to test your programs. For detailed information on the application panels used with the Foreground Option, see [Debugging Panels \(see page 19\)](#) , and for detailed information concerning the commands used with the Foreground Option, see [Debugging Commands \(see page 40\)](#) .

- [Preparation for Test Execution \(see page 119\)](#)
- [Execute a Program Until an Error Occurs \(see page 120\)](#)
- [Stop a Looping Program \(see page 121\)](#)
- [Trace a Program in Reverse \(see page 121\)](#)
- [Stop a Program at a Particular Statement \(see page 121\)](#)
- [Stop at a Statement and Step by Verb \(see page 122\)](#)
- [Stop a Program When a Subscript Overflows \(see page 123\)](#)

Preparation for Test Execution

Use the following steps to compile a program to be tested under the control of the application and to begin foreground test execution.

If you are testing an IMS program, first read this section, then read Test IMS Applications. Also, if you are testing a DB2 application, see [Test IBM DB2 Applications \(see page 133\)](#). If you are testing an application that uses any database, batch link is the suggested method for debugging.

1. Compile the program that you wish to test using the procedure set up by your installer. This procedure has a post-compile step that generates the symbolic information needed to debug your program. Alternatively, you can do the following:
 - Add the symbolic information using the Utilities panel if you have the listing available.
 - Take advantage of the dynamic symbolic support feature, if you have CA Endevor SCM r7 SP2 or later installed.
2. Log in to TSO and enter the TSO command PROFILE WTPMSG. This command causes all messages that normally appear in the batch job log (write-to-programmer messages) to be sent to the terminal.
If you place the PROFILE WTPMSG in the application CLIST, you will not need to reenter it each time you invoke the application.
3. Invoke the application (ask your technical support personnel for instructions, if needed).
If you are testing an IMS or DL/I program, also review the instructions given in the section Testing IMS Applications.
If you are testing under CA Roscoe, also review the instructions in the section Testing Under CA Roscoe.
4. When the Primary Option Menu displays, select Option 1: Foreground.

5. Fill in the relevant information on the Execution Control panel. Enter the name of the main program of the application that you want to debug, the execution parameters, if there are any, and the STEPLIB to be used for this test. If your execution JCL was converted to ALIB format, you can specify that ALIB data set and member here. Then all the information, described previously will be extracted from the ALIB file and the DDs that are required for execution will also be allocated. If you do not have an ALIB for this application, you will need to allocate any required DDs through the TSO ALLOC command or through Option 3: Allocate off of the Primary Option Menu.

If you are testing an IMS or DL/I program, also review the instructions given in the section Testing IMS Applications.

6. Fill in the name of the program that you wish to debug in the Monitored Programs section of the Monitor Control panel. The name of the program is the PROTSYM member name for the program to be tested. Also fill in the data set name of the Symbolic file, the PROTSYM that was used when the program was compiled and post-processed. When all the information has been filled in, press Enter.

If any of the programs specified on the Monitor Control panel are Visual Age PL/I for OS/390 programs and you entered a Y in the Monitor PL/I field, the PL/I PROTSYM/Load Module Map panel displays. Fill in the load module names for any PL/I programs that you want to debug. If you are using CA Endeavor SCM to manage your load module and listing libraries, you can activate the dynamic symbolic support feature (DSS) by designating a PROTSYM file as the receiver for DSS post processor-created symbolic files. This reduces your pre-testing setup time and ensures you will always debug with the correct symbolic file associated with the load module being monitored.

The execution of the application begins.

7. If no errors are encountered, the Initial Intercept panel displays. This is a display of the source statements beginning with the first executable statement. The title line should read *INITIAL* Intercept.
 8. The program is ready to execute the application. Enter the GO command in the Command field and press Enter, or press the PF key set to GO.
- The program executes until one of the following conditions is encountered:

- An abend occurs: The Intercept panel is displayed, highlighting the statement where the abend was detected.
- The program runs to normal completion. The Execution Control panel displays with the return code in the upper right corner indicating that the run has completed.

Execute a Program Until an Error Occurs

When an abend occurs, use application commands to correct the bad data and continue the test.

For example, consider the following COBOL statements:

```
041600    MOVE +20          TO VARIABLE-LENGTH-1.
041700    MOVE FIELD1       TO SALARY-TO-DATE.
041800    MOVE +30          TO VARIABLE-LENGTH-3.
```

If FIELD1 contains invalid numeric data such as spaces, an SOC7 occurs when an attempt is made to execute statement 041700. When the SOC7 occurs, the Intercept panel displays: the abend reason code appears on the title line and statement 041700 is highlighted.

To check the contents of FIELD1, use the DISPLAY command to format and display WORKING STORAGE. To change the contents of FIELD1, move the cursor under the data displayed at the right of the data item, and key in the new value for the data item.

Press the END key to return to the Intercept panel. Then issue the GO command to resume the test session.

Stop a Looping Program

Press the Attention key or PA1 key to stop a looping program. This causes the debugger to stop execution at the next executed statement. The Attention key works for terminals that are attached to the system through an SNA controller. The PA1 key works for terminals attached to the system using other means. To have the PA1 key work on most terminals, you must first press the RESET key.

If you cannot locate the Attention or RESET and PA1 keys, ask your technical support person for assistance.



Note: If you are attempting to break from a looping program and the PA1 key does not work, press RESET and then the PA1 key.

Trace a Program in Reverse

A program can be stepped backward from the point of an abend to determine the cause of the abend condition.

Use the PREV command to trace the program's execution in reverse. The PREV command does not affect the contents of the application's variables. It also does not change the location where the GO command will resume execution.

To view the statement that has been executed, enter P or PREV in the Command field of the Intercept panel and press Enter.

Stop a Program at a Particular Statement

Stop a program at selected statements by setting unconditional breakpoints at those statements. From the Intercept panel, locate the statement where the program will be stopped, then use the U line command or the UNCOND command to set a "before" unconditional breakpoint that stops *before* the statement is executed. You can also use the) line command or UNCOND command to set an "after" unconditional breakpoint that stops *after* the statement is executed. [To access the Intercept panel, review the Preparation for Test Execution section.](#)

For example, consider the following COBOL statements:

```
041400 0000-INITIALIZATION SECTION.
041500
041600     MOVE +20 TO VARIABLE-LENGTH-1.
041700     MOVE +25 TO VARIABLE-LENGTH-2.

041800     MOVE +30 TO VARIABLE-LENGTH-3.
```

The program is stopped at the first line of this code so that you can examine the value of the VARIABLE-LENGTH variables used in the next three lines.

Set the breakpoint by typing a U or a) in the left margin where the breakpoint will be set. In this example, type a U at statement 41400 and press Enter. Type GO and press Enter.

You can also use the UNCOND command to set the breakpoint. It is especially useful if you want to automatically execute commands at the breakpoint. Assume that VARIABLE-LENGTH-4 was a data-item that should have been initialized in this paragraph. The MOVE +45 TO VARIABLE-LENGTH-4 statement was omitted from the program source code. Add the statement by setting a breakpoint and specifying the SET command on the UNCOND command.

In this example, set the breakpoint using either of the following UNCOND command formats:

```
UNCOND 41400 (SET VARIABLE-LENGTH-4 = +45)
      or
UNCOND 0000-INITIALIZATION (SET VARIABLE-LENGTH-4 = +45)
```

Press Enter to set the breakpoint and then issue the GO command to execute the program. When the program begins execution of statement 041400, the Intercept panel displays. The title line reads BREAKPOINT INTERCEPT and line 041400 is highlighted. The Breakpoint Intercept panel displays before the statement is executed. The assignment of +45 to VARIABLE-LENGTH-4 has been done automatically.

Stop at a Statement and Step by Verb

Step through a section of code by setting unconditional breakpoints at particular statements and then use the STEP and GO commands to execute each verb. For example, consider the following statements:

```
049400  DIVIDE CONV-YEAR BY 4
049500      GIVING CONV-LEAP-COUNT
049600      REMAINDER CONV-LEAP-REMAINDER.
049700  ADD +1 TO CONV-LEAP-COUNT.
049800  IF CONV-LEAP-REMAINDER IS EQUAL TO ZERO
049900      THEN
050000      SUBTRACT +1 FROM CONV-LEAP-COUNT.
050100
050200  IF CONV-DAY IS NOT EQUAL TO ZERO
050300      THEN
050400      SUBTRACT +1 FROM CONV-DAY.
050500
050600  COMPUTE CONV-TOD =
050700      ((((((CONV-YEAR * 365) + CONV-LEAP-COUNT + CONV-DAY)
050800      * 24) + CONV-TIME-HOUR) * 60) * CONV-TIME-MINUTE)
050900      * 60) + CONV-TIME-SST.
051000
```

Assume that one of the IF clauses in the previous example is not functioning as expected. Step through this section of the code by using the following command sequence:

UNCOND Sets a breakpoint at statement 049400, the DIVIDE verb.	
49400	
GO or	Executes the program until statement 049400 is reached; then the Intercept panel
GO key	displays. The title line reads UNCOND BEFORE INTERCEPT. Statement 049400 is highlighted.
STEP 1	Executes (or steps) one verb each time you enter the GO command.

GO or GO key	Executes one verb and redisplay the Intercept panel. This occurs each time you enter GO, until the step count is turned off. This process lets you monitor the flow of control through the program.
-----------------	---

Assume that statement 050600 has been reached and you want to execute the program until this loop is reentered without entering GO commands for each statement. To reset the program to normal execution, you must turn off the step count and enter the GO command once more:

STEP 0 Turns off the step count (sets it to zero).

GO or GO key	Executes the program normally until statement 049400 is reached again. The Intercept panel displays, the title line reads UNCOND BEFORE INTERCEPT. Statement 049400 is highlighted.
-----------------	---



Note: As an alternative to using the STEP 1 command in conjunction with the GO command, you can use the NEXT command. This command executes up to the next verb.

Stop a Program When a Subscript Overflows

Many SOC4 abends are caused by a subscript exceeding its expected range (overflowing). This causes unrelated data in a program to be overlaid or garbled. The application can stop a program when a particular data item contains or exceeds a value or another data item. This is done by setting a conditional breakpoint using the WHEN command.

For example, assume that subscript SUBS1 is exceeding its range (1 through 30). To stop the program at the statement where SUBS1 exceeds 30, enter the following command:

```
WHEN SUBSCRIP SUBS1 GT 30
```

This command stops the program when SUBS1 is greater than 30. The character string SUBSCRIP in the command is the name assigned to the WHEN condition. It is displayed in the title of the W when-name Intercept panel whenever this condition is encountered; for example, the title of the Intercept panel reads:

```
WHEN SUBSCRIP BEFORE INTERCEPT
```

The when-name identifies which WHEN condition was encountered. The when-name can also be a statement number. For example, assume that SUBS1 is updated in statement 15200 and that SUBS1 exceeds its range.

Enter the WHEN condition as follows:

```
WHEN 15200 SUBS1 GT 30
```

This way the condition will only be checked when the specified statement number is being executed.

Test IMS Applications

If you are licensed for the IMS Option, you can use the product to test and debug either batch or online IMS applications. Test the IMS programs with the product the same way as other programs, but remember that the IMS program is not the first program that is executed when an IMS application is invoked; the IMS program is attached by the IMS Region Controller (DFSRRRC00). This occurs whether the application is a DB/DC application (MPP or BMP) or a DB application (batch).

- [IMS Batch Program Testing \(see page 124\)](#)
- [Batch Procedures for BMP Testing \(see page 126\)](#)
- [Test Procedure Using BTS \(see page 130\)](#)
- [BTS Interactions \(see page 131\)](#)
- [Test COBOL Programs Which Run Under an ADF Shell \(see page 132\)](#)

Your product interfaces with the IBM BTS (Batch Terminal Simulator) program, which you use to simulate online applications.



Notes:

- The Region Controller is a module, DFSRRRC00. It is not the IMS control region. The IMS control region is an MVS address space that runs the program DFSRRRC00. The application runs in an MVS address space, TSO that can also run the program DFSRRRC00. Neither TSO nor CA InterTest Batch runs in the IMS Control Region.
- The instructions in this section are for debugging an application under Foreground, Option 1 on the Primary Option Menu. The suggested method for testing IMS applications is using Option 5, Batch Link. [For more information about batch link, see Batch Link Facility \(see page 155\).](#)
- A facility exists for debugging your online IMS application without the use of BTS. [For more information about debugging IMS DC applications, see Batch Link Facility \(see page 155\).](#)

IMS Batch Program Testing

The JCL for executing a batch IMS program has an EXEC statement that specifies the module DFSRRRC00. The application program name is specified as a parameter to the DFSRRRC00 module. To facilitate testing IMS batch applications with your product, create a CLIST to allocate IMS and CA InterTest Batch-related data sets. Ask your technical support person for assistance in creating the CLIST, if needed.

The CLIST allocations are as follows:

- **IMS-related:**

```
DFSRESLB  IMS  IEFRDER  DFSVSAMP
```



Note: On the ddname DFSRESLB, the IMS RESLIB data set is concatenated to itself. This is necessary to prevent a dynamic allocation to the ddname. If the IMS RESLIB data set is not concatenated to itself under the ddname DFSRESLB, USER 0684 abends and other unpredictable user abends can occur.

▪ **CA InterTest Batch-related:**

INT1PARM INT1LOAD INT1PNNL INT1PROF INT1MSGL INT1CLOG INT1CLIB INT1REPT



Note: For additional information about CA InterTest Batch-related CLISTS, see [Basic Foreground Demo Session \(https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session\)](https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session).

Preparation for Test Execution - Batch Programs

Replace Step 5 in the section Preparation for Testing Execution with the following information:

- On the Execution Control panel, specify the name of the IMS RESLIB data set as the load library.
- The first program to be executed is always DFSRRC00.
- The execution parameters are the same as the parameters used in a batch execution.

Generally, the execution parameters are DLI,membername,psbname. If the application program module name and the psbname are the same, you do not need to specify the psbname parameter.

- The application requires a STEPLIB DD statement that identifies the load library containing the application program to be executed.
- You must also ALLOCATE the data bases that are used by the IMS program that will be tested.

The application generates a dynamic STEPLIB DD statement from the data set names that are listed under the STEPLIB portion of the Execution Control panel. [For more information about dynamic STEPLIB facility in Execution Control Panel, see Debugging Panels \(see page 19\).](#)

Use the Monitor Control panel to identify the programs that have their execution controlled by the application. Identify only the programs that you want to debug with the application. Also, provide the PROTSYM files that contain the symbolic information for the programs that you would like to debug.

Example:

In this example assume that you have two driver modules to debug and test: DLIPGM1 and DLIPGM2. DLIPGM1 calls five subprograms: SUBPGM1, SUBPGM2, SUBPGM3, SUBPGMA, and SUBPGMB. DLIPGM2 calls SUBPGMA and SUBPGMB. SUBPGMA and SUBPGMB do not need to be debugged. In this run, they are executed, but not symbolically debugged. Thus, their names do not need to be included on the Monitor Control panel.

```
----- CA InterTest Batch MONITOR CONTROL Panel -----
COMMAND ==>
                                                    Monitor PL/I ==> N
-----Monitored Programs -----
```

```

==> DLIPGM1      ==> SUBPGM1      ==> SUBPGM2      ==> SUBPGM3 ==> DLIPGM2
==>              ==>              ==>              ==>              ==>
==>              ==>              ==>              ==>              ==>
==>              ==>              ==>              ==>              ==>
==>              ==>              ==>              ==>              ==>

-----Symbolic (PROTSYM) Files -----Endevor Auto Populate
==> 'USER01.PROTSYM'
==>
==>
==>
==>
==>
==>
==>
==> Y
==> Always
==> Auto-Populate
==> Non-LE-Enabled
==> Assembler?
==> ==> Y (Y/N)
==>
==>

```

Batch Procedures for BMP Testing

BMPs are batch IMS programs that run under the control of an online IMS control region. The job is executed in an IMS BMP region. The BMP accesses databases that are allocated to IMS. Thus, the CLIST you use to invoke the application will not include allocation statements for the databases.

If you need assistance in creating the CLIST, ask your technical support person.



Note: If you prematurely end your test session with an END, CANCEL, or QUIT command when testing BMPs, the BMP region does not know the testing session is ending. The BMP region can abend with an S13E.

The following screen shows you an example of an Execution Control panel for IMS BMP testing:

```

----- CA InterTest Batch EXECUTION CONTROL Panel -----
COMMAND ==>

----- Allocations -----

ALIB Dsname  ==>
      Member  ==>

EXEC Job Step ==>          Proc Step ==>

----- Execution Overrides -----

PGM  ==> DFSRRC00          Initial Commands ==>

PARM ==> BMP,PAYROLL,,,N0003,,2,,,0002,,,IMST
      or Number of Linkage Parameters ==>

Task Libraries ==> 'IMS.RESLIB'
      DSNNAME      ==> 'USER01.LOAD'
      or           ==>
      (DDNAME)     ==>

```

From the Monitor Control panel, the process to test a BMP is the same as the process to test an IMS batch program.

Define Your Application

At least one BMP must be added to the IMS/VS Stage 1.

Ask the appropriate technical staff member to add the following application definitions to your IMS/VS System:

1. The APPLCTN MACRO must be defined with the DOPT option so that a new copy of the PSB is obtained each time that the program executes.
2. The PARMLIM must be zero. The option PGMTYPE=BATCH forces the PARMLIM to zero.
3. Define a different BMP for each applications programmer who is testing in this manner. Each programmer should be assigned a unique BMP to avoid having one programmer overwriting the PSB of another.

Note: You should only use this on an IMS/VS test system. The DOPT option can have a performance impact on a production IMS/VS system.

Test Your Application

The BMP defined is used to test the online application. Perform the following steps:

1. Replace the PSB for the BMP in the dynamic PSBLIB with the application PSB using the BMP PSB name.
For example, the BMP PSB is ABCD1234 and the application PSB is MYAPPLTN. Thus, the PSB defining MYAPPLTN should be linked into the dynamic PSB library with a name of ABCD1234. The programmer can use the same BMP name for any IMS/VS online applications to be tested and debugged.



Note: The dynamic PSBLIB data set must be concatenated after the ACBLIB data set. The PSB must reside in a library with the same format as IMSVS.ACBLIB and be concatenated to the IMSACB DD statement. The dynamic PSB must not be a member of IMSVS.ACBLIB.

2. Log on to the IMS/VS test system.
3. Enter an IMS/VS SET command in the form:

```
/SET TRAN trancode
```

where *trancode* is the trancode of the BMP defined in the previous section. This command directs all input messages from this LTERM (logical terminal) to *trancode*. Normally MFS provides the trancode to IMS/VS. The trancode of the application to be tested is different from the *trancode* through which the application Batch gains control.

4. Enter as many transactions as necessary. The transactions remain queued on the IMS/VS message queue until a test session is initiated.

5. Log off IMS/VS. You might have a VTAM product, which lets you do a VTAM switch without logging off IMS/VS.
6. Log on to TSO.
7. Begin a session with the following entries on the Execution Control panel:
 - a. The name of the first module to be executed is DFSRRC00
 - b. The execution parameters for the first program are as follows:
 BMP,mbr,psb,in,,ostd,,stimer,,,cputime,,,insid
 where:
 - mbr-The name of the application program to be tested
 - psb-Psbnname of the BMP defined previously containing the PSB of the application program being tested
 - in-The trancode of the BMP defined previously
 - ostd-&opt&spie&test&dirca
 - &opt - Action to be taken if the batch message region starts and no control program is active. Default is N (ask operator for decision).
 - &spie - Turn on the SPIE option (0) or not (1)
 - &test - Check (1) or not (0) the addresses in the user call list for validity
 - &dirca - Required for dynamic PSBs
 - stimer-Whether to set the timer (1) or not (0). Must be set to 1 if CPUTIME 0
 - cputime-BMP task timing option. No timing (0) or maximum task time (*n*) where *n* is from 1 to 1440 minutes
 - insid-The IMS/VS system identifier where you want the BMP to execute
8. End the session, after testing and debugging with the application.
9. Log off TSO.
10. Log on to IMS/VS to the LTERM used in Step 3.
11. Review the output messages that may have been sent to your LTERM from your application program.
12. Enter the IMS/VS RESET command as follows:
 /RESET

Application Testing Scenario

If you have two programmers testing online IMS/VS applications, you could define the following MACROS:

```
APPLCTN DOPT,PSB=INT01,PGMTYPE=BATCH
TRANSACT CODE=(INT01A),PRTY=(0,0)
APPLCTN DOPT,PSB=INT02,PGMTYPE=BATCH
TRANSACT CODE=(INT02A),PRTY=(0,0)
```

(PRTY=(0,0) is the default when PGMTYPE=BATCH.)

Jones would like to test and debug program PAYROLL while Smith tests program MANUFACT. Jones would either do a PSBGEN or copy his PSB (PAYROLL) into the dynamic PSBLIB member INT01. Likewise, Smith would do the same thing for his PSB (MANUFACT) but as member INT02.

Jones logs on to IMS/VS. He enters:

```
/SET TRAN INT01A
```

He then enters his FORMAT and data to execute his transaction. He can enter as many transactions as he likes. When he is done, he may either log off of IMS/VS or do a VTAM switch. He then logs on to TSO to start his test session. On the Execution Control panel under EXECUTION PARAMETERS he enters the following:

```
PARM ==> BMP,PAYROLL,INT01,INT01A,,N00003,,2,,,0002,,,IMST
```

When he has ended the application, he can log back on to IMS/VS and view any output messages that his application generated to his LTERM. Once he has viewed all of the IMS/VS output screens, he enters the following command before logging off IMS/VS:

```
/RESET.
```

Smith logs on to IMS/VS. He enters the following command:

```
/SET TRAN INT02A
```

He then enters his FORMAT and data to execute his transaction. He can enter as many transactions as he likes. When he is done, he can either log off IMS/VS or do a VTAM switch. He then logs on to TSO to start his test session. On the Execution Control panel under EXECUTION PARAMETERS he enters the following:

```
PARM==> BMP,MANUFACT,INT02,INT02,,N00003,,2,,,0002,,,IMST
```

When he has ended the application, he can log back on to IMS/VS and view any output messages that his application sent to his LTERM. Once he has viewed all of the IMS/VS output screens, he enters the following command before logging off IMS/VS:

```
/RESET
```

Since IMS/VS sees the program with a trancode other than the trancode of the program being tested, different users may schedule the program into multiple IMS/VS regions.

Test Message Switching Applications

Testing an IMS/VS application whose input messages are placed on the message queue by another IMS/VS program requires a modification to the procedure outlined in the section Testing Your Application. Remember that you use the /SET TRAN trancode command on your IMS/VS terminal to

direct IMS to queue all transactions that are entered from your LTERM to the trancode specified in the SET command. Therefore, your application running as BMP transaction *trancode* is able to retrieve any messages on the queue destined for your application.



Note: The suggested method for testing IMS applications is using Option 5, Batch Link.

Unfortunately, any messages that are inserted to the message queue destined for another program are not tagged with the BMP trancode. There is a way to work around this situation, as follows:

1. Follow steps 1 through 7 in the section Testing Your Application.
2. Set a breakpoint before the call to insert the message on the message queue. This is usually after the CHNG call.
3. Display the contents of the parms passed to your program using the DI or LI command.
4. Modify the trancode on the message to be inserted to be the BMP trancode and return to the Intercept panel.
5. Continue the testing of the application.
6. Return to the Execution Control panel, split the screen, and select Option 6 (TSO COMMAND PROCESSOR).
7. Copy the PSB for the program that retrieves the inserted messages into the dynamic PSBLIB data set with the BMP PSB member name. End the split screen mode.
8. Change the second execution parameter, mbr, to the name of the program retrieving the inserted messages, on the Execution Control panel.
9. Begin testing the application. [Follow steps 8 through 12 in the section Testing Your Application.](#)

Test Procedure Using BTS

Like an IMS batch application, a DB/DC application executed using BTS is not the first program executed. The module BTSTSOST is the first module executed. It in turn attaches the IMS region controller, DFSRRC00, which attaches the application program to be executed. To facilitate testing of IMS applications with both CA InterTest Batch and BTS, create a CLIST to allocate IMS, BTS, and CA InterTest Batch-related data sets. If you need help in creating the CLIST, ask your technical support staff for assistance.

The CLIST allocations are given as follows:

IMS-related:

DFSRESLB IMS IEFORDER DFSVSAMP

BTS-related:

BTSIN BTSOUT BTSPUNCH BTSSNAP QIOPCB QALTPCB QALTRAN TASKLIB

CA InterTest Batch-related:

INT1PARM INT1LOAD INT1PNNL INT1PROF INT1MSGL INT1CLOG INT1CLIB INT1REPT



Note: For additional information about CA InterTest Batch-related CLISTs, see [Basic Foreground Demo Session \(https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session\)](https://docops.ca.com/display/CAITSD11/Basic+Foreground+Demo+Session).

Preparation for Test Execution - BTS

Replace **Step 5** in the section Preparation for Test Execution with the following step:

- On the Execution Control panel, specify the name of the load library containing the BTS load modules. The first program to be executed is BTSTSOST. The execution parameters are the same parameters that are specified to BTS. Generally, the parameter that is specified is **DLI,,,**. If you specify the program member name, the psbname, or both, as you would to test a batch IMS program, a USER 0642 abend can occur (parm exceeds maximum length). The programs and the PSBnames to be tested are identified on ./T cards in the BTSIN data set.



Note: The dynamic STEPLIB function is performed by the BTS Tasklib option. Do not specify a TASKLIB ddname on the Execution Control panel, or BTS will believe that TASKLIB ddname is not allocated and terminate. Allocate the ddname TASKLIB in the CLIST to invoke the application. The load libraries containing the modules to be tested must be allocated to the ddname TASKLIB.

Use the Monitor Control panel to identify the programs that have their execution controlled by the application. Identify only the programs that you want to debug. Also, provide the PROTSYM files that contain the symbolic information for the programs that you would like to debug.

BTS Interactions

After you have entered the Execution Control panel and the Monitor Control panel, BTS begins execution by reading the BTSIN data set. A BTS0007I message and images of the BTS input commands are displayed on the terminal. BTS requests a BTS command, a /FORMAT statement, or a /*. The /FOR modname causes BTS to display a formatted IMS screen for input. Once the data has been input, BTS attaches the application program. CA InterTest Batch gains control. When the application program calls CBLTDLI, BTS again has control until the IMS language interface module passes control back to the calling program. As a result, the BTS displays are interleaved with the Intercept panels.

If the BTS session ends prematurely and there was no BTS error message displayed on the terminal, review the BTSOUT data set. The BTSOUT data set often contains data that does not appear on the terminal.



Note: The application intercepts all keyed PA1s during the BTS testing session.

Message Switching

Message switching is when an IMS application program inserts a message to the message queue that is either an output message to another IMS terminal or an input message to another transaction. If the message is an input message to another transaction, then BTS schedules another program to process the input message. To set breakpoints in the IMS program that processes the message inserted into the message queue, perform the following steps:

1. Program A inserts a message for program B. When the first executable statement in program A displays on the terminal, enter **QUALIFY B** on the command line. The beginning of the listing for program B displays. Set any UNCOND or WHEN breakpoints that you want. Enter **QUALIFY A** on the command line to return to the program A display.
2. You can also accomplish this by specifying UNCOND ALL ENTRY on the command line of the Intercept panel upon initial entry into the first IMS application program.

BTS0067I MESSAGE

If this message should occur while testing in a BTS environment, ensure the BTSEXT parameter has been added to the PARMLIB member used to invoke the application.

Display and Figure Verbs in a BTS Environment

When using these verbs in a BTS message switching environment, ensure the BTSSYSO parameter has been added to the PARMLIB member used to invoke the application. For more information, see [Applications Using Message Switching and DISPLAY or Figure Verbs in CA InterTest Batch Options \(https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options\)](https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options) . Without this option, abends OC4 and/or OC1 can occur.

Test COBOL Programs Which Run Under an ADF Shell

Use this product to test COBOL programs that run under an ADF *shell*. The recommended procedure for doing this is the same as for testing COBOL programs under BTS. A programmer specifies BTSTSOST on the Execution Control panel as shown in the previous section and the names of the COBOL programs to be tested on the Monitor Control panel. [Both of these panels are shown in the previous section Preparation for Test Execution - BTS.](#) Specify the four-character ID of the ADF SIGN-ON screen on the /FOR statement. All of the functions of the application are available while testing the specified COBOL programs.

Although this application does not display or intercept code that is written in the ADF Audit Language, some users use the application facilities to help debug their ADF code. One way is to insert a call to a dummy COBOL program at whatever point in the ADF code they want to intercept execution. Once in the Breakpoint Intercept panel, you can use the Option 2: CORE to display ADF work areas and tables. This requires a good knowledge of ADF. The best source for this information is the IBM IMS ADF II Manuals.

Test IBM DB2 Applications

To test IBM DB2 applications using your product in foreground, you can modify the TSO CLIST CAMRCDB2, which invokes the application. You find CAMRCDB2 in the CAVHCLS0 data set. A sample *batch* execution of DB2 programs follows to give you some background into the DB2 commands DSN and RUN and how they relate to DB2 foreground execution.



Note: The instructions in this section are for debugging an application under Foreground, Option 1 on the Primary Option Menu. The suggested method for testing DB2 applications is using Option 5, Batch Link. For more information on batch link and about debugging your DB2 Stored Procedure, see [Batch Link Facility](#) (see page 155).



Note: If you are using the Call Attach Facility of DB2, you may use the CAMRCLST CLIST to debug your application in foreground.

To run the DB2 program DSN8BC3 under CA InterTest Batch

1. Modify the TSO CLIST CAMRCDB2 by replacing INTBATCH with the data set prefix defined at install time.
2. Verify that the application's ISPF Panel library is concatenated to the IBM ISPF panel library, ddname *ISPPLIB*, and the message library is concatenated to the IBM ISPF message library, ddname *ISPMLIB*.



Note: For more information, see CAMRCDB2 in [Customize CLISTs in CA InterTest Batch Options](#) (<https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options>)

To test a DB2 program (PAYROLL in this example), perform the following steps:

1. EXEC the modified application CLIST.
The DB2 panel displays.
2. Complete the three fields on the CA InterTest Batch DB2 panel. Each field is defined in the following table:

Field	Description
DB2 Subsystem Name ==>	The name of the DB2 Subsystem where the program being tested runs.
Plan Name ==>	The DB2 Plan name for the program being tested.
Library Name ==>	The Load Library name where the program being tested resides.

1. Press Enter after completing the panel fields. The Primary Option Menu displays.
2. Select Option 1: Foreground. The Execution Control panel displays.
3. Supply the main application program name and the required STEPLIB to execute the application. Press Enter to see the Monitor Control panel.
4. On the Monitor Control panel, type in the names of the programs that you want to debug and DSNs of the PROTSYM files containing the symbolic information for the programs you want to debug. Press Enter to receive the *INITIAL* Intercept panel.
5. Run the test session in the usual way.
Notice that once you invoke CA InterTest Batch, there is no difference in the way the application is used to test DB2 programs.



Note: This procedure does not let you end the program and return to the Execution Control panel to retest the same program. You must exit the application and rerun the CLIST (Step 1).

Use the following approach when debugging in foreground with DB2 and CA InterTest Batch:

1. Get CA InterTest Batch working for non-DB2 programs.
2. Get the DB2 program running under native TSO.
3. Ensure that the DB2 program is correctly prepared as follows:
 - a. DB2 pre-processed
 - b. Compiled and post processed into a PROTSYM file
 - c. Linked
 - d. A DB2 bind step or job is run for the recompiled program
 - e. All other DB2 protocol is done
4. Modify the application CLIST as explained previously.

Test Data Generation

This product assists you in generating test data or files for testing. Consider the following COBOL program:

```
000100 IDENTIFICATION DIVISION.  
000200  
000300 PROGRAM-ID.      DATAGEN.  
000400 AUTHOR.          J. DOE.  
000500 INSTALLATION.    DATACEN.
```

```

000600 REMARKS.
000700
000800
000900 ENVIRONMENT DIVISION.
000901 INPUT-OUTPUT SECTION.
000902 FILE-CONTROL.
000910 COPY OUTFILE.
001000
001100 DATA DIVISION.
001200
001300 FILE SECTION.
001400 COPY OUTREC.
001500
001600 PROCEDURE DIVISION.
001700 BEGIN.
001800     OPEN OUTPUT OUT-FILE.
001900     0100-WRITE-IT.
002000     WRITE OUT-RECORD.
002010     GO TO 0100-WRITE-IT.
002100     0100-END-IT.
002200     CLOSE OUT-FILE.
002300     GOBACK.GOBACK Key

```

Assume that the COPY codes OUTFILE and OUTREC contain the SELECT and FD information respectively. Use the following command to set a breakpoint at the write statement and display the record contents in COBOL format:

```
UNCOND 0100-WRITE-IT (RECORDS)
```

Each time statement 1900 is executed, the COBOL program stops and a formatted display of the record for OUT-FILE appears. To generate the data, type over the values that you want, press the END key to return to the Intercept panel, and press the GO key to write the record.

Adding another File Definition, OPEN, and a READ before the WRITE could make this into a COBOL format record editor.

Test Under CA Roscoe

If you run this product under CA Roscoe, you should be aware of the following points:

- The suggested method for testing applications with CA Roscoe is using Option 5, Batch Link.
- The application runs under the ETSO OR BTSO facility of CA Roscoe.
- A special JCL conversion facility to simplify file allocations is available for CA Roscoe users and is described in [JCL Conversion for CA Roscoe \(see page 151\)](#).
- Your application PF key assignments can differ from your CA Roscoe PF key assignments. Define PF keys by selecting the KEYS option on the Primary Option Menu.
- The application lets you use the split-screen facility to establish a second session using the PF SPLIT key. You also can use standard CA Roscoe commands to initiate a second CA Roscoe session.
- Use the CA Roscoe SUSPEND function to suspend a CA InterTest Batch session and return to CA Roscoe. You can then perform other CA Roscoe functions; however, you cannot call another ETSO OR BTSO application. Resume ETSO OR BTSO to return to your session.

- Use the PF PRINT key to print to a CA InterTest Batch data set, which can be allocated to the AWS or any MVS data set.
- Use the CA Roscoe print facility to print to a CA Roscoe data set.

Allocations Facility for ISPF

Before you can begin a debugging session, you must allocate the data sets that your program will be using. The allocations facility for TSO/ISPF users provides a number of features that makes this set-up process easier.

- [The Allocations Main Menu \(see page 137\)](#)
- [Option 1 ALIB \(Allocations Library\) Functions \(see page 137\)](#)
- [Option 2 The JCL Library/Allocations Functions \(see page 145\)](#)
- [Option 3 Current Allocations \(see page 149\)](#)
- [Option 4 JCL Conversion/Allocation Options \(see page 150\)](#)

Briefly, the allocations facility lets you to do the following actions:

- Create and manage ALIB (Allocations Library) allocations members. ALIB is the internal allocations format for the application. The ALIB is a PDS that consists of members created through the ALIB editor or JCL converter of this application. You can create new allocations library members, or edit, browse, allocate and free data sets specified in existing members. Once you save an ALIB, you can use its allocations in the future, saving you valuable debugging set-up time.



Note: The ALIB dataset is optional and is used by the CA InterTest Batch foreground testing facility. For more information, see [Allocate ALIBs \(Optional\) in CA InterTest Batch Options \(https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options\)](#).

- Convert allocation lists from ALIB to CLIST format. Although ALIB is an efficient format that is sufficient for most situations, conversion to CLIST is useful in complex allocation situations that are beyond the straightforward free-and-allocate capabilities of ALIB (conditional data set allocations, for example). Converting to CLIST may also be necessary if your particular operating environment requires the CLIST format.
- Browse existing JCL members in PDS, CA Librarian, and CA Panvalet libraries.
- Convert JCL to ALIB format. This feature is useful if you have existing JCL that you would like to use for your allocations. The system's conversion utility fully supports JCL allocation features, and converted JCL is displayed in Edit mode for further processing. You can also use the resulting ALIB on the Execution Control panel to allocate and free the required application DDs. You can convert JCL to CLIST format by first converting it to ALIB format, then converting it from ALIB to CLIST (it is not necessary to save in ALIB format). Using this feature saves you the time needed to key in and debug ALIB or CLIST statements from scratch.

The allocations facility panels conform to ISPF file name and command conventions (for example, the ISPF Locate command).



Note: For more information on how to use the JCL conversion facility, CA Roscoe users should see [JCL Conversion for CA Roscoe \(see page 151\)](#)

The Allocations Main Menu

Access the Allocations Main Menu by selecting option 3 from the Primary Option Menu. The Allocations Main Menu the starting point for accessing all of the allocations facility's features:

The fields on the Allocations Main Menu are as follows:

1 ALIB	Create new allocations members, edit and browse existing members, allocate/free data sets specified in members, and convert ALIB members into CLIST format.
2 JCL	Browse existing JCL members in PDS, CA Librarian, or CA Panvalet libraries. You can also select members for conversion from JCL to ALIB format. The converted JCL is displayed in Edit mode for further processing.
3 CURRENT	Display current session allocations and free, browse, or edit allocated data sets.
4 OPTIONS	Specify various JCL conversion options and ALIB member export options, and identify procedure libraries (proclibs) used for expanding JCL procedures found during JCL conversion.

Option 1 ALIB (Allocations Library) Functions

The ALIB library option lets you create new allocations members, edit and browse existing members, allocate/free data sets specified in members, export (convert) ALIB members to CLIST format, and import (convert) JCL members to ALIB format. Selecting this option displays the Allocations Library panel.

Select an ALIB Member from the Allocations Library Panel

The application displays the Allocations Library panel if you select option 1 (ALIB) from the Allocations Main Menu. Use this panel to enter the name of the member you want to edit.

For ISPF library members: The Allocation Library panel has the following input fields:

For ISPF library members:

Project, Group, Type	Required. Enter the three-level qualifier for standard data set names in ISPF.
Member	Optional. If you do not know the member name, leave this field blank or enter a member name pattern. Valid ISPF <i>wild card</i> member pattern characters are:
	% Substitutes for any one character. May be the first character.

* Substitutes for any number of characters. May not be the first character.

For PDS members:

Data set name	Optional. If your member is not in an ISPF library, enter the actual, cataloged name of the file. The name can be up to 44 characters long, with segments up to eight characters long, separated by periods. You can specify any MVS data set name and use wild card substitutions, as described for the Member field.
Volume serial	Optional. This field identifies the VOLSER of the data set specified in the data set name field and is necessary only when a file is not cataloged.

To select an ALIB Member from the Allocations Library panel:

1. Enter the ISPF library or PDS information for the member you want to edit.
2. Press Enter.

If you entered a complete member name, the application displays the Allocations Edit panel. If you entered a partial member name, the application displays the ALIB Member Selection List.

Select an ALIB Member from the ALIB Member Selection List

If you did not specify a complete member name on the Allocations Library panel, the application displays the ALIB Member Selection List panel. This panel lists all the allocations library members or those that match the pattern.

```
CA InterTest Batch  ALIB: USER01.INTBATCH.ALIB -----
Command ==>
Cmd Name      Message  VV.MM Created   Modified   Size  Init  Mods ID
NEW           01.02 14.060   14.077   12:47 00090 00090 00000 USER01
PGM1          01.08 92.351   14.077   12:33 00001 00001 00000 USER01
PGM2          01.01 14.077   14.077   14:09 00013 00013 00000 USER01
PGM3          01.08 14.062   14.077   10:27 00002 00002 00000 USER01
TEST1         01.01 14.077   14.077   14:10 00002 00002 00000 USER01
TEST2         01.02 14.078   14.078   10:11 00001 00021 00000 USER01
```

Enter the following line commands in the Cmd field before each member in the list:

-
- AL** To allocate all the data sets in the ALIB member.
-
- B** To browse the ALIB members. The Allocation List Browse panel displays. From this panel you may view the allocation list in an ALIB. You can scroll through the members and use the ISPF Locate (L) command to locate specific members. The Allocation List Browse panel is identical in format to the Allocations Edit panel, but no edits may be made.
-
- C** To display the Convert to CLIST panel, which lets you convert the selected ALIB member to CLIST format. [For more information, see Convert ALIB to CLIST \(Export ALIB\).](#)
-
- E** To edit the ALIB member. Issuing this command displays the Allocations Edit panel. [For more information, see Edit ALIB Allocations.](#)
-
- F** To free all the ddnames in the ALIB member.
-

Edit ALIB Allocations

Display the Allocations Edit panel in any one of the following ways:

- From the Allocations Library panel by specifying a particular member
- From the ALIB Member Selection List panel by entering the line commands E or S next to a member
- From the JCL Allocations Step Selection List panel by selecting converted JCL steps for allocation

The Allocations Edit panel lets you create, modify, and manipulate lists of data set allocations. A list may be saved into the ALIB, exported (converted) into a CLIST library, allocated, or freed. JCL may be imported (converted) into ALIB format.

A sample Allocations Edit Panel is shown next:

```
CA InterTest Batch ALIB EDIT: PDS USER01.INTBATCH.ALIB(PGM1) -----
COMMAND ==>                                     SCROLL==> PAGE
LINE COMMANDS - I(NNN), D(NNN), R(NNN), AL(LOCATE), F(REE), S(PACE)
PRIMARY COMMANDS - ALLOCATE, SAVE, CREATE, RESET, CANCEL, IMPORT, EXPORT,
REPLACE, FREE, OPTS, END.  FOR AN EXPLANATION, SEE THE USER GUIDE OR HIT HELP.
CMD DDNAME DATA SET NAME                      DISP VOLSER DCB AL
MSG> Imported from USER01.JCL(PGM1)
MSG> USER01  //USER01R  JOB (42400000), 'USER01', CLASS=A, M
MSG> RUN      //RUN      EXEC PGM=MAINCOB, REGION=4M, PARM= '          RUN
MSG> .. MAINCOB
.... @STEPLIB USER01.LOAD                      SHR          N
.... SYSPRINT SYSOUT=X                          NEW          N
.... SYSABOUT SYSOUT=X                         NEW          N
.... SYSOUT    SYSOUT=X                         NEW          N   A
.... SYSUDUMP  SYSOUT=X                         NEW          N
.... CARDIN    USER01.CARDIN.FILE               NEW          Y
.... CARDOUT   USER01.SYSOUT                     SHR          N
```

CMD	Enter all line commands in this area.
DDName	Data definition name. Enter a valid ddname in this field.
Data Set Name	Data set name. Enter a valid data set name in this field. Data set names must be fully qualified and without apostrophes. SYSOUT= and TERMINAL are valid entries.
Disp	The disposition of the data set. Enter the disposition of the data set in this field. Valid dispositions are: OLD SHR NEW Default: SHR.
Volser	The old volume serial, if any, of the data set. You may enter another volume serial on which the data set is to reside.
DCB	Yes DCB information is present No There is no DCB information for the data set
AL	A The ddname is allocated blank The ddname is not allocated

Primary Commands

The Allocations Edit Panel primary commands, which you enter on the COMMAND ===> line, are shown next.

ALLOCa	Allocates all the allocations listed on the screen.
SAVE	Saves any changes made to the list of allocations in the current ALIB member.
CREate	Writes the list of allocations to a new ALIB member. Issuing this command displays the Copy to Allocation Library panel.
RESet	Removes all non-relevant MSG lines created during JCL conversion or during allocation.
CANcel	Exits without saving any changes and redisplay the Member Selection List panel from which you entered the editor. All changes to the member that were not saved are discarded.
IMPort	Invokes the JCL Converter to convert raw JCL into ALIB format. Import JCL from a PDS, CA Librarian, CA Panvalet, and current session allocations libraries. The application displays the JCL Import Specifications panel. For more information, see Converting JCL to ALIB (Import JCL).
EXPort	Converts the current ALIB member into CLIST format. For more information, see Convert ALIB to CLIST (Export ALIB).
REPlace	Replaces a member with the contents of the current member. Issuing this command displays the Copy to Allocation Library panel.
END	Ends the current Edit session and displays the Member Selection List panel from which you entered the editor. The member is automatically saved if it was modified.
FREE	Frees all ddnames in the current member.
OPTS	Lets you change conversion and allocation options. Invoking this command displays the JCL Conversion/Allocation Options panel. For more information, see Option 4: JCL Conversion /Allocation Options.

Line Commands

Enter line commands in the Cmd field next to the ddname:

Dataset Attributes Panel

Use the Dataset Attributes panel to review and change the data set attributes including DCB information, space information, and extended (SMS) attributes. This panel displays when you select a ddname on the Allocations Edit panel.

Dataset Attributes Panel: Fields

The following table describes the Dataset Attributes Panel fields:

LRECL	The logical record length of the data set.
BLKSIZE	The block size of the data set.
RECFM	The record format of the data set. Valid formats are:
	F Fixed

	FB Fixed blocked FBA Fixed blocked with carriage control character V Variable VB Variable blocked U Undefined Default: the current record format of the data set.
DSORG	The organization of the data set. Valid organizations are: PO Partitioned Organization (PDS) PS Sequential
SPACE UNIT	The space unit used by the data set for primary and secondary storage. Valid space units are: BLK Blocks TRK Tracks CYL Cylinders Default: the current space unit of the data set.
PRIME SIZE	The primary storage.
SECONDARY	The secondary storage.
DIRECTORY BLOCKS	The number of directory blocks.
RELEASE	Whether unused storage should be released. Valid values are: Yes Release unused storage No Do not release unused storage Default: No
UNIT NAME	The unit name.
VOLUME SERIAL	If you entered a VOLSER on the Allocations Edit panel, it is displayed here. If this field is blank or you want to change the VOLSER, enter the new VOLSER.
OR ALLOCATE LIKE DDNAME OR DSNAME	Use this field to obtain DCB information from another data set or ddname. Enter a valid ddname in parentheses (for example,INT1LOAD), or a fully qualified data set name.
STORCLAS	The SMS storage class.
DATACLAS	The SMS data class.
MGMTCLAS	The SMS management class.
DSNTYPE	One of the following valid data set types:

	LIBRARY PDF PIPE HFS
RECORD	One of the following valid record organizations: KS Keyed sequential file ES Entry-sequenced file RR Relative record file LS Linear space file
AVGREC	Allocate space by records using one of these valid specifications: U Unit is one record T Unit is one thousand records M Unit is one million records
KEYOFF	Identifies the offset within the record where the key resides, for keyed files only.
KEYLEN	Identifies the length of the key.

To change DCB information

1. Enter the DCB information in the appropriate fields.
2. Enter **END** on the command line and press Enter, or press the End key.

Result: The application saves the DCB information and redisplay the Allocations Edit panel.



Note: Enter **CANCEL** to ignore the DCB information and to redisplay the Allocations Edit panel.

Copy to Allocation Library Panel

The application displays the Copy to Allocation Library panel whenever the ALIB editor needs the name of the ALIB data set and member in order to write the allocations list. The application displays this panel if you issue a Replace or Create command from the Allocations Edit panel.

To specify the ALIB data set name

Note: The ALIB data set is optional. If your installation is not using an ALIB data set no entry is required on this panel. Press PF3 several times until you return to the CA InterTest Batch PRIMARY OPTION MENU. For more information, see the section [Allocate ALIBs \(Optional\)](https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options) in [CA InterTest Batch Options](https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options) (<https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options>).

1. If the data set to which you are writing is an ISPF library, complete the information for the ISPF Library fields (Project, Group, Type, and Member). For more information on these fields, see [Select an ALIB Member from the Allocations Library Panel](#).
2. If the data set to which you are writing is not an ISPF library, complete the information for the Other partitioned data set fields (Data set name, Volume serial). For more information on these fields, see [Select an ALIB Member from the Allocations Library Panel](#).
3. Enter **Yes** in the Replace like-named member field if you want the editor to overwrite members by the same name in the library (the default is No).
4. Press Enter.

Result: The application creates or replaces the ALIB data set and redisplay the Allocations Edit panel.

Convert ALIB to CLIST (Export ALIB)

Access the Convert to CLIST Options panel in one of these ways:

- From the ALIB Member Selection List panel by entering the line command C (Convert to CLIST) next to the member to convert.
- From the Allocations Edit panel by entering the primary command EXP (Export) on the command line.

If your allocation situation is too complex for the limitations of the ALIB format (for instance, if you need conditional allocations or need to use other TSO commands beyond simple free-and-allocate statements), you may find it useful to translate your ALIB statements to CLIST.

The ISPF library and Other partitioned data set fields, in which you specify your destination (*TO*) data set, are the same as those on the Allocations Library panel.

The other fields are as follows:

Replace Determines if the target CLIST library is to be replaced.
?

Yes Replace the target CLIST library

No Do not overwrite the existing CLIST library. After you press Enter, the cursor is placed in the Data Set Name field and you can specify a new name

Edit CLIST Determines if the ISPF editor will be invoked to edit the converted member.
?

Yes The ISPF editor will be invoked to edit the converted member

No The ISPF editor will not be invoked

To convert from ALIB to CLIST

1. Specify the destination (TO) data set in either the ISPF library or the Other partitioned data set fields. Modify the member name, if necessary.
2. Enter **Yes** or **No** in the Replace? and Edit CLIST? fields.
3. Press Enter.

Result: The application performs the conversion. If you entered Yes in the Edit CLIST field, the application invokes the ISPF editor to allow you to further modify the CLIST. If you entered No in the Edit CLIST field, the application redisplay the panel from which the converter was invoked.

Convert JCL to ALIB (Import JCL)

The application displays the JCL Import Specifications panel when you use the Import command on the Allocations Edit Panel. Use this panel to enter the name and type of the JCL library member from which you want to import JCL for conversion.

The ISPF library and Other data set fields are the same as those on the Allocations Library panel. The other fields are as follows.

Option	The data set format of the data set you want to import for conversion.
Blank	The member is in a PDS.
L	The member is in a CA Librarian library.
P	The member is in a CA Panvalet library.
S	Import the currently allocated data set list (current session allocations). No other fields on this panel need to be completed if you choose this option.
CA Librarian index type	When converting CA Librarian members, determines the amount of directory information displayed.
Q	Quick. Displays the member name only and takes less time than the S (Short) option.
S	Short. Displays more information on the selected member than the Q (Quick) option.

To convert from JCL to ALIB

1. Enter the data set format of the JCL member you want to import for conversion.
2. In either the ISPF library or the other partitioned data set fields, specify the source (*FROM*) data set that contains the JCL you want to import for conversion.
3. Press Enter.

Result: If you have requested the JCL Conversion/Allocations Options Panel, it will be displayed next. Otherwise, the JCL member selection panel is displayed.

Option 2 The JCL Library/Allocations Functions

Option 2 (JCL) on the Allocations Main Menu panel lets you select existing JCL members for conversion to ALIB format. PDS, CA Librarian, and CA Panvalet libraries are supported. Use the resulting ALIB member on the Execution Control Panel to allocate and free DDs.

Specify the JCL to Convert

The application displays the JCL Library panel when you select the JCL option from the Allocations Main Menu. Use this panel to specify the name and type of the JCL member you want to convert.

Complete the following input fields:

Option	The format of the data set you want to import for conversion.
Blank	The member is in a PDS.
L	The member is in a CA Librarian library.
P	The member is in a CA Panvalet library.

For ISPF Library members:

Project, Group, Type	Required. Enter the three-level qualifier for standard data set names in ISPF
Member	Optional. If you do not know the member name, leave this field blank or enter a member name pattern. Valid ISPF <i>wild card</i> member pattern characters are:
%	Substitutes for any one character. May be the first character.
*	Substitutes for any number of characters. May not be the first character

For PDS members:

Data set name	Optional. If your member is not in an ISPF library, enter the actual, cataloged name of the file. The name can be up to 44 characters long, with segments up to eight characters long, separated by periods. You can specify any MVS data set name. You can use wild card substitutions, as described for the Member field.
Volume serial	Optional. This field identifies the VOLSER of the data set specified in the Data set name field and is necessary only when a file is not cataloged.
CA Librarian index type	Used only when converting CA Librarian files. Determines the amount of directory information the application displays.
Q	Quick. Displays the member name only and takes less time than the S (Short) option.

S Short. Displays more information on the selected member than the Q (Quick) option.

To specify a JCL member for conversion:

1. Enter the ISPF library or PDS information.
2. If you are converting a CA Librarian member, enter the CA Librarian index type you desire.
3. Press Enter.
4. The application displays the JCL Member Selection List panel.

Use the JCL Member Selection List

After specifying a partial or complete member name on the JCL Library panel, the application displays the JCL Member Selection List panel.

```
CA InterTest Batch JCL: PDS USER01.INTBATCH.JCL-----
Command ==>                                     Scroll ==> PAGE
Enter B to Browse member, S to select member for JCL conversion
Cmd Name      Message  VV.MM Created  Modified  Size  Init  Mods ID
ABSDUMP              01.00 14.325   14.325   15:47 00006 00006 00000 USER01
ASMHAL              01.00 14.154   14.154   11:40 00015 00015 00000 USER01
TESTX               01.02 14.008   14.211   15:47 00043 00022 00027 USER01
TEST2               01.03 14.188   14.062   09:13 00005 00005 00000 USER01
TEST3               01.00 14.188   14.188   10:34 00007 00007 00000 USER01
```

This panel displays the JCL members that match the member name or wild card pattern specified in the Member field on the JCL Library panel. If you made no entry in the Member field, the application displays all the members of the library.

Enter the following line commands in the Cmd field before each member in the list:

B To browse the JCL library member. the application displays the JCL Browse panel.

S To select the JCL member for conversion to ALIB format. Once the JCL conversion has ended, the application displays the JCL Allocations Step Selection List panel. Note that if you have not set the option to suppress the Conversion/Allocation Options panel, it is displayed before converting the JCL and displaying the JCL Allocations Step Selections Panel. [For more information, see Option 4: JCL Conversion/Allocation Options.](#)

The JCL Browse Panel

The application displays the JCL Browse panel when you use the Browse command on the JCL Member Selection List panel.

Use the JCL Browse panel to check the contents of a member selected for conversion and to determine the STEP names. Scroll through the member to find the STEPs you want to use. Entering an END command or pressing the End key redisplay the JCL Member Selection List panel.

```
CA InterTest Batch Browse JCL: PDS USER01.INTBATCH.JCL -----
Command ==>                                     Scroll ==> PAGE
Cmd ----- JCLs -----
//USER01C JOB (12300000,76),'RUN',CLASS=A,MSGCLASS=Z,
//  NOTIFY=USER01
//PROCLIB DD DISP=SHR,DSN=USER01.PROCLIB
//QDD1 EXEC COB,N=QDD
//COB.SYSIN DD DISP=SHR,DSN=USER01.QDD.COBOL
```

```
//QEE2      EXEC  COB,N=UXX
//COB.SYSIN DD  DISP=SHR,DSN=SYS2.ROSCOE56.DAILY.BKUP(-5)
```

Select Converted JCL Steps for Allocation

The application always displays the JCL Allocation Step Selection List panel after JCL conversion, if there were multiple steps in the JCL. The panel lists all JCL EXEC cards in the member:

```
CA InterTest Batch JCL: PDS USER01.INTBATCH.JCL(TEST3) -----
Command ==>                               Scroll ==> PAGE
Enter S beside step to select, or press END key to select all
Cmd----- Steps -----
DELETE //DELETE EXEC PGM=IEFBR14
C //C EXEC PGM=IKFCBL00,REGION=4M,
LKED //LKED EXEC PGM=IEWL,
PRINTIT //PRINTIT EXEC PGM=IEBGENER,REGION=1024K
DELETE //DELETE EXEC PGM=IEFBR14
C //C EXEC PGM=IKFCBL00,REGION=4M,
LKED //LKED EXEC PGM=IEWL,
PRINTIT //PRINTIT EXEC PGM=IEBGENER,REGION=1024K
```



Note: If you have not set the option to suppress the Conversion/Allocation Options panel, it displays before converting the JCL and displaying the JCL Allocations Step Selections Panel. For more information, see [Option4: JCL Conversion/Allocation Options](#).

To select converted JCL steps for allocation

1. On the JCL Allocation Step Selection List panel, enter **S** next to the steps whose data set allocations you want to use or enter the END command (or press the END key) to use all the steps. The steps you select form your allocation list.
2. Press Enter (unless you used an END key instead of entering the END command in step 1).
3. The application displays the Allocations Edit panel so you may edit the newly created Allocation List, saved in an allocation library or converted to CLIST format.

For more information on working with allocation libraries, see [Edit ALIB Allocations](#).

The Conversion Process

When you select JCL for conversion to ALIB format, the JCL converter checks the JCL for syntax. If the JCL executes procedures and PROCLIBs were specified on the Conversion/Allocation Options panel, the JCL converter expands the procedures found in the PROCLIBs, substitutes parameters, and performs step overrides. If the JCL executes a procedure that is not found in a PROCLIB, a WARNING panel is displayed and JCLCheck message CAY6027E is included in the ALIB member, indicating the name of the procedure that was not found.

```
CA InterTest Batch ALIB EDIT: PDS CAI.CAVHJCL(DEMOJCL) -----
COMMAND ==>                               SCROLL==> PAGE
LINE COMMANDS - I(NNN), D(NNN), R(NNN), AL(LOCATE), F(REE), S(PACE)
PRIMARY COMMANDS - ALLOCATE, SAVE, CREATE, RESET, CANCEL, IMPORT, EXPORT,
REPLACE, FREE, OPTS, END. FOR AN EXPLANATION, SEE THE USER GUIDE OR HIT HELP.
CMD DDNAME DATA SET NAME DISP VOLSER DCB AL
MSG> Imported from CAI.CAVHJCL(DEMOJCL)
MSG> DEMOJCL //DEMOJCL JOB (124400000), 'INTBAT10 DEMOJCL
MSG> STEP1 //STEP1 EXEC PGM=CAMRC0BB,REGION=4M STEP1
```

```

MSG> ... .CA
.... @STEPLIB USER01.LOAD                SHR          N
.... SYSPRINT SYSOUT=X                   NEW           N
.... SYSOUT   SYSOUT=X                   NEW           N
.... REPORT   SYSOUT=X                   NEW           N
***** BOTTOM OF DATA *****

```

To save the created Alib member

1. Place the cursor on the command line and type in 'SAVE' to save the ALIB version of the JCL.
The Copy to Allocation Library screen opens.
2. Fill in the library and member name where your ALIB member is to be saved and press Enter.
The ALIB member is saved.

Special Considerations

There are some special conversion considerations for certain types of DD statements. The application allocates DD statements to a particular TSO session by using SVC 99, which performs dynamic allocation. SVC 99 does not allow the following four DD statements to be dynamically allocated:

- STEPLIB
- JOBLIB
- STEPCAT
- JOBCAT

The application changes STEPLIB to @STEPLIB and JOBLIB to @JOBLIB.

JCL Keywords Supported for Conversion

The JCL converter supports the following JCL keywords and positional operands. The degree of converter support is noted.

JCL Keywords	Description
DSN=	Fully supported.
DISP=	All operands except conditional dispositions are supported. If a DD with a conditional disposition is passed, the third sub parameter is ignored.
DCB=	The DCB keywords supported are: RECFM= BLKSIZE= LRECL= DSORG=
HOLD=	Ignored.
SYSOUT=	Fully supported. Only the CLASS parameter is used.
UNIT=	Fully supported.
VOL=	Fully supported. Only the VOLSER parameter is used.

JCL Keywords	Description
SUBSYS=	Fully supported. Check with the vendor to make sure that a DD can be allocated for this subsystem with SVC 99.
SPACE=	Fully supported.
STORCLAS=	Fully supported.
DATACLAS=	Fully supported.
MGMTCLAS=	Fully supported.
DSNTYPE=	Fully supported.
RECORG=	Fully supported.
AVGREC=	Fully supported.
KEYOFF=	Fully supported.
KEYLEN=	Fully supported.
LIKE=	Fully supported.
REFDD=	Fully supported.

JCL Keywords Not Supported for Conversion

The JCL converter does not convert all JCL keywords because many parameters in batch JCL are not supported in TSO CLISTs. The converter ignores backward references and keywords that it does not convert.

AMORG=	FLASH=
BURST=	FREE=
CHARS=	LABEL=
CHKPT=	MODIFY=
CNTL=	MSVGP=
COPIES=	OUTLIM=
DDNAME=	OUTPUT=
DEST=	PROTECT=YES
DLM=	QNAME=
DSID=	UCS=
FCB=	

Option 3 Current Allocations

The application displays the Current Allocations panel when you select option 3 (CURRENT) from the Allocations Main Menu panel. The current allocations option displays current session allocations and lets you free, browse, or edit allocated data sets.

The Current Allocations panel lists currently allocated data sets, which may have been previously allocated in TSO Logon Procedures (such as STEPLIB), by a CLIST, or by the Allocation Facility. A sample Current Allocations List panel follows:

```
----- CA InterTest Batch Current Allocations List -----
Command ==>
Line commands: F - Free, B - Browse, E -Edit.
Cm DDName Disp Dataset Name Volser
ISP10155 SHR USER01.INTBATCH.ISPMLIB OSI011
ISPPROF SHR USER01.ISPF.ISPPROF CAI801
SYSPRINT TERM
SYSTEM TERM
SYSOUT TERM
SYSUDUMP JES
SYSIN TERM
SYS00001 SHR SYS1.BROADCAST MVSLIB
ISPALIB SHR ISP.ISPALIB MVXAD1
ISRCFIL SHR ISR.ISPFILMF.CFIL MVS003
SYSHelp SHR SYS2.HELP MVSLIB
SHR SYS1.HELP MVXAD1
SHR USER.HELP CAI800
ISPTABL SHR USER01.ISPF.ISPPROF CAI801
ICQTABL SHR ICQ.ICQTLIB MVXAD1
SYSPROC SHR SYS2.CLIST MVSLIB
SHR ICQ.ICQCCLIB MVXAD1
SHR ICQ.ICQACLIB MVXAD1
SHR ISR.ISRCLIB MVXAD1
SHR SYS1.RMFCLS MVXAD1
```

You may scroll through the list and enter the following line commands next to any data set in the list:

F/S Free the data set/ddname.

B Browse the data set. This is an ISPF function and operates only if it is supported by your ISPF installation.

E Edit the data set. This is an ISPF function, and operates only if it is supported by your ISPF installation.

Option 4 JCL Conversion/Allocation Options

The application displays the JCL Conversion/Allocation Options panel when you select option 4 (OPTIONS) from the Allocations Main Menu panel.

Use the Options option to specify JCL conversion and ALIB member export options and Proclibs (procedure libraries) used for expanding JCL procedures found during JCL conversion.

The Current Allocations/Options fields are described next:

Proclib 1-7 The procedure library names you want to use for expanding procedures found in JCL during conversion. When your JCL job stream executes procedures, assign the procedures a Proclib to be used during the conversion process. Enter up to seven procedure library names. If you require more than seven Proclibs, use a pre-allocated ddname instead.

Pre-allocated DD If you require more than seven Proclibs, or if your Proclibs are pre-allocated by other means (for example, in the TSO logon procedures or custom CLISTs), enter the name of a pre-allocated ddname here.

EasyProclib	For EasyProclib users, this field specifies whether the JCL conversion facility uses private JCL procedure libraries identified by the //PROCLIB ddname in the job stream.
Yes	The JCL conversion facility uses private JCL procedures using the //PROCLIB ddname in the job stream.
No	The JCL conversion facility will not use private JCL procedures using the //PROCLIB ddname in the job stream.
Default: No	
CLIST 'Control'	Whatever you enter here appears in the CLIST control statement when you convert from ALIB to CLIST (for example, NOMSG and NOLIST). For more information on valid CLIST control statements, see the IBM publication <i>TSO Command Language Reference Manual</i> .
DCB info on (Attr, Alloc)	Determines whether DCB information appears on the attribute statement or the allocate statement in your CLIST when you convert from ALIB to CLIST. The choice is mainly dictated by user-preference.
ALLOC	Put DCB information on the generated ALLOC statement.
ATTR	Generate DCB information on a separate ATTRIB statement.
Default: Attr	
If allocated (Reuse, Free)	Controls the allocation process during CLIST conversion. If the data set is already allocated:
R	The application tries to reuse the allocation. The REUSE statement is generated on the allocations card.
F	The application frees the data set before allocation. The FREE statement is generated on the allocations card.
Default: Free	
If exists (Delete, Use - for new data sets only)	Used for allocating a new, non-temporary data set. Generates a DELETE statement in the CLIST to prevent a <i>Duplicate Data Set Name</i> error.
DELETE	Generates a DELETE statement in the CLIST. This deletes any existing data set allocated with the same name.
USE	A <i>Duplicate Data Set Name</i> error is generated if there is an existing data set allocated with the same name.
Default: USE	
Suppress (Yes or No)	Indicates whether or not this panel should be suppressed when converting JCL. Once this option is set to Yes, this panel only displays when choosing option 4 from the Allocations Main Menu.

JCL Conversion for CA Roscoe

This section describes how to use the JCL Conversion facility for CA Roscoe.

The JCL Conversion facility lets you convert JCL to CA Roscoe RPF format so that CA Roscoe can execute it. Use this facility to convert the JCL that performs the allocations required by programs you are testing in foreground with your product.

- [Access the JCL Conversion Facility \(see page 152\)](#)
- [Specify the JCL Member for Conversion \(see page 152\)](#)
- [Convert the JCL Member \(see page 154\)](#)
- [Review and Edit the Converted JCL \(see page 154\)](#)
- [Execute the Converted JCL \(see page 155\)](#)
- [Convert JCL to ALIB in Batch \(see page 155\)](#)

Access the JCL Conversion Facility

To enter the JCL Conversion facility, enter the following command on the command line:

```
[xxx.]IBCONV
```

where xxx is the prefix of the library in which the product is installed. Omit this prefix if the application is installed in a common execution library.

The application displays the JCL Converter panel.

Specify the JCL Member for Conversion

The JCL Converter panel lets you specify the JCL member for conversion:

```
----- CA InterTest Batch JCL Converter -----
Option ==>
Select an Option and press ENTER
      A - AWS....
      L - LIB....
      D - DSN.... USER01.LIB.CNTL(TEST*)
      J - JOB....

      X   - Exit

JCL Conversion PROCLIBs:
      Proclib1... USER01.PROCLIB
      Proclib2... SYS2.PROCLIB
      Proclib3...
      Proclib4...
      Proclib5...
      Proclib6...
      Proclib7...
```

Identify Where the JCL Member Is Stored

JCL members are stored in one of the following areas:

AWS	CA Roscoe active working storage
LIB	CA Roscoe user library
DSN	Data set (PDS, sequential or CA Librarian)

JOB The currently attached job. You cannot change this jobname.

The information specified in these fields reflects your profile. If you change this information, for example if you change the DSN, application updates your profile, which is stored in your library in member ZZIBOPTS. Follow standard CA Roscoe conventions for specifying the AWS, Library or DSN.

Identify the PROCLIBs

In addition to specifying where the JCL member is stored, you must specify at least one PROCLIB. Each PROCLIB identifies a library where JCL procedures are stored. Even if the JCL member does not use a procedure, one PROCLIB must be specified on the JCL Conversion panel.

Display the JCL Member

Follow these steps:

1. Enter a letter (A,L,D,J) in the Option field to indicate where the JCL member is stored.
2. Complete the information for the option you selected:
 - For A (AWS), enter the AWS name or leave this field blank.
 - For L (library) or D (DSN), enter the library or DSN name or a partial name using the generic characters + or *, or leave this field blank.
 - For J (job), the name of the currently attached job already appears on the JCL Conversion panel and cannot be changed.
3. Enter at least one PROCLIB name in the PROCLIB list.
4. Press Enter.

If you entered a complete AWS, library or DSN name, CA Roscoe displays the JCL member you selected. If you entered a partial name or left the name blank, CA Roscoe prompts you for additional information. See *Selecting the JCL Member from a List*, which follows, for instructions on specifying that information. If you selected J (job), CA Roscoe displays file 2 (the JCL file) of the currently attached job.

Select the JCL Member from a List

If you do not specify a complete AWS, library or DSN name on the JCL Converter panel, CA Roscoe prompts you for additional information.

For *AWS*, CA Roscoe displays a list of members. Enter **a** to the left of the desired AWS and press Enter to attach and display the member.

For a *library*:

- If you left the library field blank on the JCL Converter panel, CA Roscoe displays the Library Facility panel. Specify the library information and, optionally, the member name, and attach the library.
- If you specify the member name, CA Roscoe displays the member.

- If you omit the member name, CA Roscoe displays the member list.
- If you specified a partial library name on the JCL Converter panel, CA Roscoe displays a list of members that meet the criteria.

When CA Roscoe displays the member list, enter **a** to the left of the library JCL member and press Enter to attach and display a member.

For a *DSN*:

- If you left the DSN field blank on the JCL Converter panel, CA Roscoe displays the Data Set Facility panel. Specify the data set information and attach the data set to display a list of members.
- If you specified a partial data set name on the JCL Converter panel, CA Roscoe displays a list of members that meet the criteria.

When CA Roscoe displays the member list, enter **a** to the left of the data set JCL member and press Enter to attach and display a member.

Convert the JCL Member

When the application displays the JCL member you specified on the JCL Converter panel or selected from a list, you can review the member to make sure it is the one you want to convert.

Follow these steps:

1. Enter **GO** on the command line.
2. Press Enter.

The application converts the JCL member to CA Roscoe RPF format and displays the following message:

RPF generated in AWS xxxx

Where xxxx is the current AWS.

After the JCL member has been converted, enter **a** to attach and display the converted JCL member.

Review and Edit the Converted JCL

You can review and, if desired, edit the converted JCL to delete unnecessary steps. Note the following:

- The converted JCL is divided into steps.
- **::** denotes a comment that identifies a STEP in the original JCL and displays the EXEC card.
- **:** denotes a message generated during conversion. These messages can be ignored.
- Each step contains:

- FREE statements to deallocate previously allocated data sets.
- DELETE statements, if necessary, to delete non-temporary data sets that need to be created and may already exist.
- ALLOC statements to allocate and, if necessary, create the data sets.

Use standard CA Roscoe editing commands to edit the converted JCL. When you finish editing the member, save it in your own CA Roscoe library.

Execute the Converted JCL

After the JCL member has been saved in your CA Roscoe library, you can execute the converted JCL member at any time to perform the allocations. Before executing a program that you want to test with the application, execute the appropriate RPF member to allocate the files required by the program.

Convert JCL to ALIB in Batch

A utility exists, CAMRAUTL, which takes a given JCL member of a PDS and creates an ALIB member from it. Review JCL member CAMRAUTL in CAVHPROC. There are three required DDs, shown in the following table:

DDname	Description
JCL	DD pointing to the JCL member to be used in creating the ALIB.
ALIB	DD pointing to the ALIB PDS.
SYSPROC	DD pointing to the PROCLIB concatenation.

JCL example:

```
// JOB
//*****
//CRE8ALIB EXEC PGM=CAMRAUTL
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//SYSPROC DD DISP=SHR,DSN=USER.JCL
//JCL DD DISP=SHR,DSN=USER.JCL(TESTJCL)
//ALIB DD DISP=SHR,DSN=USER.ALIB
//
```

Batch Link Facility

Batch link lets you debug applications from a terminal while they execute in batch.

The batch link facility offers the following advantages over standard foreground debugging:

- No foreground allocations
Executing an application in foreground requires that all data sets used by the application be first allocated in foreground. While the application provides a JCL conversion and allocations facility

for this purpose, the allocation and de-allocation process can complicate the debugging process. Applications debugged using batch link are executing in batch, so no foreground allocations are required.

- **Execution of leading and trailing steps**
Sometimes an application is executed as part of a multi-step job stream and one or more leading steps must be executed before the application can run. Batch link allows debugging of any job step executing in batch, and therefore permits the normal completion of leading steps prior to debugging. After debugging, any trailing steps can be completed prior to job termination.
- **Use of temporary files**
Because batch link permits the execution of leading steps, applications can use temporary files created by a prior step. The job stream can then delete these files when they are no longer needed.
- **Multi-step debugging**
Batch link supports the debugging of multiple steps within the same job stream. When debugging of any step completes, the job continues as usual with the execution of the following step until end of job. Each batch link user controls which steps will be debugged within a job stream before the job is submitted.
- **Multi-user debugging**
At any time during a batch link debugging session, you can suspend the session and disconnect the terminal from the batch debugging session. This lets the debugging session be retrieved by another (or even the same) user at a later time. This facilitates the transfer of the debugging session from one user to another without terminating and restarting the application.

CA InterTest Batch JCL Requirements

Before debugging a batch application using the batch link facility, modify the application JCL to enable the batch link environment. Do this manually as described here, or automatically using the batch link JCL conversion facility.

Change the application JCL for each step to be debugged. These changes include the following steps:

1. Modify the PGM= field on the // EXEC statement to execute CAMRBL01.
2. Insert the application executable library at the end of the STEPLIB or JOBLIB concatenation if it does not exist in LINKLIST.



Note: It is possible to use the dynamic symbolic support feature if you have multiple CA Endevor SCM C1DEFLT5 and are debugging as if you have a single C1DEFLT5. To do so, insert a fully qualified data set name containing the C1DEFLT5 with your site under the STEPLIB concatenation ID.

3. Insert additional DD statements for the application data sets used by the debugging engine, including INT1PARM, INT1PROF, INT1PNLL, and INT1MSG1.



Note: For more information about these additional DD statements, see The Batch Link DD Statements in [Batch Link JCL Edit Panel \(see page 164\)](#).

4. Insert an INT1OPTS DD containing in-stream debugging options, such as the original program name from the PGM= field on the // EXEC statement.



Note: For more information about valid options, see Batch Link Options in [Batch Link JCL Edit Panel \(see page 164\)](#) . Different requirements apply for DB2 applications .

The following code is a sample JCL stream for batch link:

```
// JOB
// EXEC PGM=CAMRBL01,PARM='application parms'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
// DD DISP=SHR,DSN=USER.LOADLIB
// DD DISP=SHR,DSN=USER.C1DEFLT5
//INT1PARM DD DISP=SHR,DSN=CAI.CAVHSAMP
//INT1PROF DD DISP=SHR,DSN=CAI.PROFLIB
//INT1PNLL DD DISP=SHR,DSN=CAI.CAVHPNL1
//INT1MSG1 DD UNIT=SYSDA,SPACE=(TRK,(1,1))
//INT1OPTS DD *
// EXEC=CAMRC0BB,PROFILE=USER01
/*
```

Debug a Batch Application

When a job step that has been modified for batch link executes, the program CAMRBL01 starts the debugging engine in batch. At this point the job step becomes available for debugging.

Batch link debugging sessions can only be connected to a terminal using the Batch Link Option Menu, described later in this section. You choose a batch link job step for debugging from the selection list provided. Doing so connects the terminal to the batch job, creating a batch link debugging session.

Debugging is exactly the same as it is for a standard foreground testing session, with the following exceptions:

- The application and the debugging engine both execute in batch, not in foreground. The terminal acts only as a *window* for communicating with the batch link session.
- The batch link session begins with the Monitor Control panel, not an Execution Control panel.
- When the debugging session ends, the batch job continues normally with the next step, and the terminal returns to the Batch Link Option Menu.

Use the Batch Link Menus

You can display the Batch Link menu by selecting option 5 from the Primary Option Menu:

The Batch Link Option menu consists of four options:

- **JCL**

Use this option to convert and submit JCL for batch link. Conversion is only required once per JCL stream, unless you alter your original JCL or decide to change which steps to debug. Alternatively, you can modify the JCL manually.

- **Batch Link**

Use this option to select a batch job for debugging. Batch link lists all jobs available for debugging. Select the job that you want to debug to connect your terminal to the batch job, creating a batch link session.

- **Edit JCL**

Use this option to edit and submit existing batch link JCL.



Note: This option is only supported under ISPF.

- **Scheduling**

Use this option to use the DB2 SP and IMS/DC scheduling feature. This feature lets you specify which programs in which regions you want the application to monitor, and lets all other programs run unmonitored. This feature lets you use the same region for debugging and for running the normal IMS/DC and DB2SP workload.

Batch Link JCL Conversion Panel

Option 1 of the Batch Link Option Menu displays the Batch Link JCL Conversion panel. Use this panel to identify the library name (and member if the library is partitioned) containing the JCL to be converted for batch link.

- [Select a JCL Member \(see page 159\)](#)
- [Batch Link Conversion Options Panel \(see page 159\)](#)
- [Select Job Steps \(see page 160\)](#)
- [Convert and Submit the JCL \(see page 160\)](#)
- [JCL Conversion Limitations \(see page 162\)](#)



Note: The batch link JCL conversion utility creates a new JCL member in the output destination of your choice.

The Batch Link Conversion panel fields are described next:

Field Name	Description
Project	The high-level qualifier for the library containing the JCL to be converted.
Library	The middle-level qualifier for the library containing the JCL to be converted.
Type	The lowest-level qualifier for the library containing the JCL to be converted.
Member	The name of the member to be converted when the library is partitioned. Leave this field blank for sequential files or to produce a member list for a PDS.
Data Set Name	If the JCL library name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the library in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Volume Serial	If the JCL library is not catalogued, use this field to enter the VOLSER of the DASD volume on which the library resides.
Dataset Password	For password-protected libraries only, enter the password in this field.

Once you have given values to all of the necessary fields on the panel, press Enter to advance to the next panel.

Select a JCL Member

If you have not provided a member name and your JCL library is partitioned, you are prompted to select a member for conversion.

```

----- CA InterTest Batch MEMBER SELECTION -----
COMMAND INPUT ==>                                SCROLL ==> HALF
NAME
***** TOP OF DATA *****
ARMYCOMP
ARMYRUN
ARMY5STP
ASM1
COMPNGO
DB2COMP
DB2DEM02
DB2DEM03
DB2RUN
s DEM0JCL
IMSBUILD
***** BOTTOM OF DATA *****

```


To select a member, place an **s** in the prefix area next to the member name and press Enter.

Batch Link Conversion Options Panel

After you identify the JCL library, use the Batch Link Conversion Options panel to specify options for the conversion.

The Batch Link Conversion Options panel fields are described next:

Field Name	Description
Proclib 1-7	The data set names of any procedure libraries required for converting the JCL.

Field Name	Description
Preallocated DD	Identifies the ddname used to pre-allocate procedure libraries required for converting the JCL.
Output JCL Lib	The name of the JCL library where the converted JCL should be written. If this field is left blank, a temporary output file is allocated. When specifying an output JCL library, use quotes for fully qualified data set names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Output Member	If a partitioned output JCL library is specified, enter the member name into which the converted JCL is to be written.
Unit	The unit field to be used if the converted JCL is being written to a temporary file.
 Note: If using a temporary file for the converted JCL, VIO must not be used.	
Profile Library	The name of the application profile library that should be used by the batch debugging engine, excluding the profile member name. Use quotes for fully qualified data set names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Profile Member	The name of the member within the profile library that should be used by the batch debugging engine.
Options	Enter additional batch link options in this field. These options are discussed in Batch Link JCL Edit Panel (see page 164) .

After you assign values to all of the necessary fields on the panel, press Enter to advance to the next panel.

Select Job Steps

If your JCL contains multiple steps, you are prompted to select job steps.

```
----- CA InterTest Batch Step Selection List -----
COMMAND INPUT ==>                                SCROLL ==> PAGE
      Select one or more of the following Steps:
      ***** TOP OF DATA *****
s    1    //STEP1    EXEC PGM=CAMRC0BB,REGION=4M
      2    //STEP2    EXEC PGM=CAMRC0B,REGION=4M
      ***** BOTTOM OF DATA *****
```

You can select any or all job steps in the selection list. To select a job step, place an **s** in the prefix area next to the entry.

To select multiple job steps, place an **s** in the prefix area next to each entry.

Press Enter to convert the JCL for batch link.

Convert and Submit the JCL

Once the JCL is converted, you are placed into EDIT, which lets you view or further modify the JCL prior to submitting it.

If you entered an output data set name, the JCL is written to that data set. If no output data set name is provided, a temporary sequential file is used.

```

EDIT          SYS00222.T093343.RA000.USER01.IBJCL.H01          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** ***** Top of Data *****
000001 //USER01A  JOB (42400000),'Demo JCL',CLASS=K,MSGCLASS=X,
000002 // NOTIFY=USER01
000003 //STEP1    EXEC PGM=CAMRBL01,REGION=4M
000004 //STEPLIB  DD DISP=SHR,DSN=CAI.CAVHLOAD
000005 //         DD DISP=SHR,DSN=USER01.LOADLIB
000006 //SYSPRINT DD SYSOUT=*
000007 //SYSOUT   DD SYSOUT=*
000008 //INT1OPTS DD *
000009 EXEC=CAMRC0BB, PROFILE=USER01
000010 /*
000011 //INT1PARM DD DISP=SHR,DSN=USER01.INTBATCH.CAVHSAMP
000012 //INT1LOAD DD DISP=SHR,DSN=USER01.INTBATCH.CAVHLOAD
000013 //INT1PNLL DD DISP=SHR,DSN=USER01.INTBATCH.CAVHPNL1
000014 //INT1MSGL DD DISP=SHR,DSN=USER01.INTBATCH.CAVHMSG0
000015 //INT1PROF DD DISP=SHR,DSN=USER01.INTBATCH.PROFLIB
000016 //INT1CLIB DD DISP=SHR,DSN=USER01.INTBATCH.CAVHSRC
000017 //INT1CLOG DD SYSOUT=*
000018 //INT1REPT DD SYSOUT=*
***** ***** Bottom of Data *****

```

For each job step selected for debugging, the following modifications are made for batch link:

1. The PGM= field on the // EXEC statement is modified to execute CAMRBL01, the batch link driver program.
2. The application executable library is added to the STEPLIB or JOBLIB concatenation.
3. Additional DD statements are inserted for the application data sets used by the debugging engine, including INT1PARM, INT1PROF, INT1PNLL, and INT1MSGL. These additional DD statements are described in detail later.
4. An INT1OPTS DD statement is added containing in-stream debugging options, such as the original program name from the PGM= field on the // EXEC statement.
5. If you are using the dynamic symbolic support feature and you have multiple C1DEFLT5, you can concatenate to the STEPLIB a fully qualified data set name containing the CA Endeavor SCM site ID associated with the load module being monitored. Alternatively, you can also let CA CCI discover the site id and populate the symbolic file for you.



Note: The requirements are slightly different for DB2 applications, as described.

When you are ready to submit the JCL, enter SUBMIT on the command line and press Enter. The job is submitted but you remain in the EDIT session. Exit the EDIT session by entering END or CANCEL on the command line and then pressing Enter.

The job begins execution and becomes available for debugging as soon as one of the selected steps begins to execute.

JCL Conversion Limitations

The following JCL constructs are not currently supported by the Batch Link JCL Conversion utility:

Partial DD overrides

Given the following JCL:

```
//PROC1    PROC
//S1       EXEC  PGM=TESTPGM
//TEST1    DD   DISP=( ,CATLG,DELETE) ,DSN=USER01.TESTDSN,
//          UNIT=DISK,SPACE=(1,(1,1))
// PEND
//MYPGM     EXEC  PROC1
//S1.TEST1 DD   UNIT=SYSDA
```

The DD TEST1 will not be generated correctly in the converted JCL.

JCL invokes a PROC that updates or adds JCL keywords to an EXEC statement.

Given the following JCL:

```
//PROC1    PROC
//S1       EXEC  PGM=TESTPGM, PARM=' ABC '
//TEST1    DD   SYSOUT=*
// PEND
//MYPGM     EXEC  PROC1, PARM=' DEF '
```

The PARM keyword on the EXEC card will not be updated.



Note: Any client hitting this limitation can still use the utility to do most of the conversion or you can edit the JCL and make the necessary changes manually.

When the application converts JCL for batch link, all the called catalogued procedures are expanded and all expanded steps from the procedures are given a stepname of just *procstepname*.

When there are COND= parameters in the original JCL with *stepname.procstepname* references, these are modified to use the *procstepname*. As a result, the reference may no longer be unique.

IF statements and DD statements with *stepname.procstepname* references are not modified and the *stepname.procstepname* reference is no longer valid. Therefore you must examine the converted JCL and modify these instances as appropriate.

Batch Link Selection Panel

Option 2 of the Batch Link Option Menu displays the Batch Link Selection panel. Use this panel to select a batch job for debugging.



Note: This panel is not accessible unless the batch link facility is initialized on your system and at least one batch link job step is ready for debugging.

A batch link debugging session is functionally identical to a foreground test session.



Note: For a detailed description of the foreground test facility, see [Debugging Commands](#) (see page 40) .

```

----- CA InterTest Batch Batch Link Selection -----
COMMAND INPUT ==>          *          *          *          *          *          *          *          *
                          *          *          *          *          *          *          *          *
#      Jobname  Stepname  Program  Owner    User    Trans    SYSID
*****
1      JOB01A   STEP1     TESTPGM  USER01                    TS001
2      JOB02A   STEP1     CAMRCOBB USER02                    TS002
3      JOB03X   RUN       PAYROLL  USER03  LINKED BY USER03  TS001
*****
*****
***** BOTTOM OF DATA *****

```

Debugging a Job Step

To select a job step for debugging, type **S** next to the entry on the selection list, and press Enter. The Monitor Control panel appears and the debugging session begins.



Note: Only one user at a time can select a batch link job step. If another user has selected a step, the entry is marked as 'Linked by *userid*'.

Suspended Jobs

When you select a batch link job step that was suspended, the debugging session resumes at the point at which the session was suspended.

To suspend a session enter the SUSPEND command after the initial debugging intercept. A session is suspended if a user ID is canceled during a debugging session.

Debugging on Foreign LPARs

You can debug a job that is executing within the SYSPLEX on a different LPAR than the one you are currently logged on to. The SYSID column displays the LPAR the job is executing on.

After you select a job that is executing on a foreign LPAR, enter your user credentials to start debugging.



Note: For more information, see [Configure Batch Link and Sysplex \(https://docops.ca.com/display/CAITSD11/Configure+Batch+Link+and+Sysplex\)](https://docops.ca.com/display/CAITSD11/Configure+Batch+Link+and+Sysplex).

Batch Link JCL Edit Panel

Option 3 of the Batch Link Option Menu displays the Batch Link Edit JCL panel. Use this panel to identify the library name (and member if the library is PDS) containing the JCL to be edited before submitting it for the batch link debugging session.

- [Select a JCL Member for Editing \(see page 164\)](#)
- [The Batch Link DD Statements \(see page 165\)](#)
- [Batch Link Options \(see page 165\)](#)

The following table describes the Batch Link JCL Edit panel fields.

Field Name	Description
Project	The high-level qualifier for the library containing the JCL to be edited.
Library	The middle-level qualifier for the library containing the JCL to be edited.
Type	The lowest-level qualifier for the library containing the JCL to be edited.
Member	The name of the member to be edited when the library is partitioned. Leave this field blank for sequential files or to produce a member list for a PDS.
Data Set Name	If the JCL library name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the library in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is appended as the high-level qualifier.
Volume Serial	If the JCL library is not catalogued, use this field to enter the VOLSER of the DASD volume on which the library resides.

Once you have given values to all of the necessary fields on the panel, press Enter to advance to the ISPF Edit panel. After editing, you can save your changes and submit the changed JCL for batch link execution.

Select a JCL Member for Editing

If you have not provided a member name and your JCL library is partitioned, you are prompted to select a member for editing.

```
----- CA InterTest Batch MEMBER SELECTION -----
COMMAND INPUT ==>                                SCROLL ==> HALF
NAME
***** TOP OF DATA *****
ARMYCOMP
ARMYRUN
ARMY5STP
ASM1
COMPNG0
DB2COMP
DB2DEM02
DB2DEM03
DB2RUN
s DEM0JCL
IMSBUILD
***** BOTTOM OF DATA *****
```

To select a member, place an **s** in the prefix area next to the member name and press Enter.


The Batch Link DD Statements



The following table describes the additional DD statements required for batch link. You must add these DD statements to each job step selected for debugging:

DDNAME	Data Set Name	Description
INT1PARM	CAI. CAVHSAMP	Contains the parameter members used by the batch debugger.
INT1LOAD	CAI.CAVHLOAD	Contains the product executables.
INT1PNLL	CAI.CAVHPNL1	Contains panel skeletons.
INT1MSGL	CAI. CAVHMSG0	Contains messages.
INT1PROF	userid.PROFLIB	Contains profile members used by the debugger to store user preferences.
INT1OPTS	n/a	Contains batch link options and control information.
(Optional DDNAMEs)		
INT1CLIB	CAI. CAVHSAMP	Contains INCLUDE members.
INT1CLOG	(SYSOUT)	Log output from the debugger.
INT1REPT	(SYSOUT)	Report output from the HIST and XSUM commands.

Batch Link Options

The following table describes the batch link options that you can specify in the INT1OPTS file for job steps selected for debugging.

Keyword	Description
AUTH	Specifies the USERID that is authorized to debug this application.
BYPASS	Use this option to prevent the batch link engine from stopping for input at the Monitor Control panel, the Initial Intercept panel, or both. Specify BYPASS=MONITOR to prevent the batch link engine from stopping at the Monitor Control panel unless an error is detected.
<div>  Note: The dynamic symbolic support feature is not supported when BYAPSS=MONITOR is specified in conjunction with PGM and PROTSYM options. </div>	
Specify BYPASS=INITIAL to prevent the batch link engine from stopping at the initial Intercept panel.	
DB2SP	Use this option to indicate that the application you are debugging is a DB2 stored procedure.
EXEC*	

Keyword	Description
	Specifies the name of the application's main program. For most batch job steps, this is the name which is replaced by CAMRBL01 in the PGM= parameter of the // EXEC statement. For some special environments such as DB2, see Batch Link Special Considerations (see page 173) .
ICMDS	Indicates the member in your INT1CLIB DD that includes application commands to be executed at the initial intercept.
	<div>  Note: If this command is used in conjunction with BYPASS(INITIAL), these commands are not executed until a breakpoint is reached. If you would like to bypass the initial intercept and would also like a set of commands to be executed at the initial intercept, do not use BYPASS(INITIAL). Instead, have GO as the last command executed at the initial intercept. </div>
IDLE=nn	Use the IDLE command force a debugging session to time out if it is idle for a specified amount of time. "nn" is the number of wall clock minutes that the debugging session will be allowed to sit idle before being timed out. When a time out occurs, WTOs will be issued which indicate that the timer threshold was exceeded and the debugging session will end. If this is not an IMSDC or a DB2SP debugging session, the jobstep will also abend with a U0999 abend.
IMSDC	Use this option to indicate that the application you are debugging is an online IMS application.
LOOP	Specifies the interval in seconds that a batch link job step is permitted to execute between responses from a terminal. If this interval is exceeded, the debugger cancels the batch link job step.
PGM	<p>Use the PGM option to specify which programs are to be debugged during the batch link session. Values specified for this option are used to complete the PROGRAM fields on the Monitor Control panel for the batch link session. You can specify only one program as a value, but the PGM option can be repeated up to 30 times per job step. If there are no PGM values specified in INT1OPTS, all of the symbolic information is retrieved from the profile member identified by the PROFILE option.</p> <p>However, if you specify PGM values in INT1OPTS, then no values are retrieved from the profile.</p> <div>  Note: If specified in conjunction with BYPASS=MONITOR and PROTSYM option, dynamic symbolic support is not supported. </div>
PROFILE	Specifies the name of the profile member that should be used by the debugging engine. You can use a unique profile member for each batch link job or step if desired, or specify your own user ID to use your personal profile member. If not specified, the default is the user ID used to submit the job.
PROTSYM	Use the PROTSYM option to specify the file that contains the symbolic information for the programs you want to debug. You can specify only one PROTSYM file as a value, but the PROTSYM option can be repeated up to seven times.

Keyword	Description
---------	-------------

	If the PGM option has not been specified, the PROTSYM option is ignored.
--	--



Note: If specified in conjunction with BYPASS=MONITOR and PGM option, dynamic symbolic support is not supported.



Note: An asterisk (*) next to a keyword indicates that the option is required on each selected step.

Specify options using the syntax:

keyword=value

You can specify more than one option per line if desired, separating the options by a space or a comma.

DB2 and IMS Schedule Menu

Option 4 of the Batch Link Option Menu displays the DB2 and IMS Schedule Menu panel.

- [Batch Link Schedule Display Panel \(see page 168\)](#)
- [Batch Link Schedule Add Panel \(see page 169\)](#)
- [Batch Link Schedule Change Panel \(see page 170\)](#)
- [Batch Link Schedule Delete Action \(see page 171\)](#)
- [Batch Link Schedule Import Panel \(see page 171\)](#)
- [Batch Link Schedule Export Panel \(see page 172\)](#)
- **DISPLAY**
Displays a [list \(see page \)](#) of existing DB2 or IMS schedules. Specify **DB2** or **IMS** in the Schedule Monitoring for field when you use this option.



Note: You can also use these lists to change [\(see page \)](#) or delete entries from the existing schedule.

- **ADD**
Allows adding a DB2 SP or IMS/DC schedule. Specify **DB2** or **IMS** in the Schedule Monitoring for field when you use this option.
- **IMPORT**
Enables importing a predefined set of schedules (DB2 SPS and IMS/DC) from an external data set.

▪ EXPORT

Enables exporting all schedules (DB2 SPS and IMS/DC) to an external data set.

Sharing Schedules between LPARs in a SYSPLEX

Scheduling is not limited to the LPAR you are currently logged on to. You can:

- Add an IMS / DB2 schedule on one LPAR which is then available on all LPARs.
- Change an IMS / DB2 schedule on one LPAR, the change propagates to all LPARs.
- Delete an IMS / DB2 schedule from one LPAR which is then deleted on all LPARs.
- Trigger an IMS /DB2 schedule and select it for debugging from any LPAR.
- Use the IMPORT and EXPORT functions for all scheduled items available on the LPARs within the SYSPLEX.



Note: For more information, see [Configure Batch Link and Sysplex \(https://docops.ca.com/display/CAITSD11/Configure+Batch+Link+and+Sysplex\)](https://docops.ca.com/display/CAITSD11/Configure+Batch+Link+and+Sysplex).

Batch Link Schedule Display Panel

Based on the database type selected in the Schedule Options panel, the following active schedule entries are displayed:

```
----- CA InterTest Batch DB2 SPS Schedule Maintenance -----
COMMAND ==> SCROLL ==> CUR
COMMAND      : A - ADD a new entry to Schedule.
LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
-----
```

CMD	REGION/JOBNAME	PROGRAM	TYPE	ONE TIME	USER	TRANSACTION
	BIMNTH01	ASSET*	DB2	YES		
	BIMNTH01	SALES*	DB2	YES		
	WKLYPAY	TIMEATTN	DB2	YES		
	WKLYPAY	FCIA	DB2	YES		
	WKLYPAY	FEDW2	DB2	YES		
	WKLYPAY	STATEW2	DB2	YES		
	WKLYPAY	CASHFLOW	DB2	YES		
	QTRLY*	GL001	DB2	YES		
	QTRYL*	ASSET001	DB2	YES		
	QTRLY*	SALES001	DB2	YES		
	QTRLY*	INVNTROY	DB2	YES		
	QTRLY*	CHAININV	DB2	YES		
	QTRLY*	WAREHSE	DB2	YES		
	QTRLY070	DSTRUTON	DB2	YES		

```
-----
***** BOTTOM OF DATA *****
```

The Batch Link IMS/DC Schedule Display panel displays IMS/DC on the first line of the panel, and IMS under the TYPE column. The rest of the display is similar to the DB2 SPS display.

```
----- CA InterTest™ IMS/DC Schedule Maintenance -----
COMMAND ==> SCROLL ==> CUR
COMMAND      : A - ADD a new entry to Schedule.
LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
```



```

-----
CMD      REGION/JOBNAME  PROGRAM  TYPE  ONE TIME  USER      TRANSACTION
*****
BIMNTH01  ASSET*    IMS     YES    IMSUSER1  IMSTRANS
BIMNTH01  SALES*    IMS     YES    USER2IMS  IMSTRX02
WKLYPAY   TIMEATTN  IMS     YES    IMSU0003  IMST0003
WKLYPAY   FCIA      IMS     YES    MPPUSER4  MPPTRX04
WKLYPAY   FEDW2     IMS     YES    USERIMS5  TESTTRAN
WKLYPAY   STATEW2   IMS     YES    USERMPP6  TRANSMPP
*****
***** BOTTOM OF DATA *****

```

Batch Link Schedule Add Panel

Select option 2 from the Schedule Options panel or enter **A** on the Schedule Display panel. The Schedule Add panel displays.

The Batch Link Schedule Add panel fields are described next:

Field Name	Description
REGION /JOBNAME	<p>This is usually required for IMS/DC scheduling.</p> <p>If program name selection is independent of region, job name, or both, leave this field blank, otherwise specify a maximum of eight character IMS region name or z/OS job name that is commonly associated with the program name to be monitored. Use the entire length of the field, including trailing blanks as selection criteria.</p> <p>A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. Thus ABC* matches any name whose first three characters start with ABC.</p> <p>An asterisk (*) as the first character of this field denotes that any region/job name is considered a match.</p>
PROGRAM NAME	<p>This is a required field. Provide a maximum of eight-character valid program name (DB2 SP name or IMS/DC name) you want the application to monitor during your debugging session.</p> <p>The entire length of the field, including trailing blanks is used as selection criteria.</p> <p>A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. Thus ABC* matches any name whose first three characters start with ABC.</p> <p>An asterisk (*) as the first character of this field denotes that any program name is considered a match.</p>
ONE TIME SCHEDULE	<p>If this option is set to Y, yes, the entry is deleted from the schedule upon selection for monitoring.</p> <p>Change this to N, no, if you do not want the entry deleted after selection.</p> <p>Note that keeping the entry in the schedule means that the product always monitors the named program in a debugging session when all of the selection criteria are met.</p>
DATABASE TYPE	<p>This field is automatically filled in based on the value you entered in Schedule Monitoring for ==> of the Schedule Options panel.</p>

Field Name	Description
USER ID	This field is used for IMS/DC only. This is the user ID assigned to the transaction submitted in the MPP region. If this value is supplied, monitoring happens only if the specified user ID starts the transaction.
TRANSACTION	This field is the IMS transaction name used for IMS/DC only. A trailing asterisk (*) denotes a generic name. Characters to the left of the asterisk are used for selection. For example ABC* matches any name whose first three characters start with ABC.

Batch Link Schedule Change Panel

To change displayed entries, enter **C** under the CMD column next to the entries you want to change:

```

----- CA InterTest Batch DB2 SPS Schedule Maintenance -----
COMMAND ==> SCROLL ==> CUR
COMMAND : A - ADD a new entry to Schedule.
LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
-----
CMD    REGION/JOBNAME  PROGRAM    TYPE    ONE TIME  USER    TRANSACTION
***** TOP OF DATA *****
      BIMNTH01        ASSET*     DB2      YES
      BIMNTH01        SALES*     DB2      YES
C      WKLYPAY         TIMEATTN   DB2      YES
      WKLYPAY         FCIA       DB2      YES
      WKLYPAY         FEDW2     DB2      YES
      WKLYPAY         STATEW2   DB2      YES
      WKLYPAY         CASHFLOW  DB2      YES
C      QTRLY*          GL001     DB2      YES
C      QTRYL*          ASSET001  DB2      YES
      QTRLY*          SALES001  DB2      YES
      QTRLY*          INVNTROY  DB2      YES
      QTRLY*          CHAININV  DB2      YES
      QTRLY*          WAREHSE   DB2      YES
      QTRLY070        DSTRUTON  DB2      YES
***** BOTTOM OF DATA *****

```

The Schedule Change panel displays once for each selection:

```

----- CA InterTest Batch DB2 SPS Schedule Change -----

CHANGE any field by overtyping the displayed value.
Press ENTER to accept the change, or END to cancel.

-----
      REGION/JOBNAME  ==> WKLYPAY
      PROGRAM NAME    ==> TIMEATTN
      ONE TIME SCHEDULE ==> Y      (Y/N)
      DATABASE TYPE    ==> DB2
      USER ID         ==>
      TRANSACTION      ==>

PFkeys: HELP - Tutorial
        END  - Exit

```

After changing the schedule and pressing Enter, the Change panel for the next marked line, if any, is displayed. This continues until all of the lines marked with C have been displayed for change.

After all the changes have been done, the system refreshes the Display panel to reflect the changes.



Note: Pressing the END key (PF3) only cancels the currently displayed change and any further Change panels queued for display. For more information about the Change panel fields, see Batch Link Schedule Add Panel.

Batch Link Schedule Delete Action

To delete an entry, enter **D** under the CMD column next to the entry you want to delete and press Enter:

```
----- CA InterTest Batch DB2 SPS Schedule Maintenance -----
COMMAND ==> SCROLL ==> CUR
COMMAND      : A - ADD a new entry to Schedule.
LINE COMMANDS: D - DELETE selected entry. C - CHANGE selected entry
-----
```

CMD	REGION/JOBNAME	PROGRAM	TYPE	ONE TIME	USER	TRANSACTION
	***** TOP OF DATA *****					
	BIMNTH01	ASSET*	DB2	YES		
	BIMNTH01	SALES*	DB2	YES		
D	WKLYPAY	TIMEATTN	DB2	YES		
	WKLYPAY	FCIA	DB2	YES		
	WKLYPAY	FEDW2	DB2	YES		
	WKLYPAY	STATEW2	DB2	YES		
	WKLYPAY	CASHFLOW	DB2	YES		
D	QTRLY*	GL001	DB2	YES		
D	QTRYL*	ASSET001	DB2	YES		
	QTRLY*	SALES001	DB2	YES		
	QTRLY*	INVNTROY	DB2	YES		
	QTRLY*	CHAININV	DB2	YES		
	QTRLY*	WAREHSE	DB2	YES		
	QTRLY070	DSTRUTON	DB2	YES		
	***** BOTTOM OF DATA *****					

The system deletes all the marked entries and refreshes the display to reflect the deletion.

Batch Link Schedule Import Panel

When you select option 3 from the Schedule Options panel, the Batch Link Schedule Import panel displays.

The Batch Link Schedule Import panel fields are described next:

Field Name	Description
Project	The high-level qualifier for the data set containing the schedule to be imported.
Library	The middle-level qualifier for the data set containing the schedule to be imported.
Type	The lowest-level qualifier for the data set containing the schedule to be imported.
Member	The name of the member to be imported when the data set is partitioned. Leave this field blank for sequential files or to produce a member selection list for a PDS.
Data Set Name	If the import data set name does not conform to PROJECT.LIBRARY.TYPE, enter the name of the data set in this field. Use quotes for fully qualified names. If quotes are omitted, your ISPF prefix is used to form the high-level qualifier.
Volume Serial	If the data set is not catalogued, use this field to enter the VOLSER of the DASD volume on which the data set resides.

Note that the DCB attributes of the IMPORT data set are RECFM=FB, LRECL=80, DSORG=PS or PO.

Alternatively you can run the schedule import function as a batch job. Use the following sample JCL to accomplish the task:

```
// JOB
//*JOBPARM SYSAFF=xxxx
//IMPORT EXEC PGM=CAMR40IP
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

Instead of doing a complete replacement of the schedule table, import can also perform an additive import when you code an execution parameter:

```
// JOB
//*JOBPARM SYSAFF=xxxx
//IMPORT EXEC PGM=CAMR40IP, PARM='ADD'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

Entries are added based on slot availability and using the same first available rules.

The following code is return code from import operations:

RC = 0 successfully imported

RC = 4 schedule table does not exist

RC = 8 INT1SKUT DD statement missing.

Batch Link Schedule Export Panel

When you select option 4 from the Schedule Options panel, the Batch Link Schedule Export panel displays.

See the Batch Link Schedule Import panel fields for a discussion of the various similarly named fields.



Note: Export creates a new PDS member if member name provided does not exist. If one does exist, it is overwritten.

Alternatively, you can run the schedule export function as a batch job. Use the following sample JCL to accomplish the task:

```
// JOB
//*JOBPARM SYSAFF=xxxx
//EXPORT EXEC PGM=CAMR40XP
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//INT1SKUT DD DISP=SHR,DSN=CAI.SCHEDULE.DATA(MEMBER)
```

The following code is return code from export operations:

RC = 0 successfully imported

RC = 4 schedule table does not exist

RC = 8 INT1SKUT DD statement missing.

Batch Link Special Considerations

- [DB2 Considerations \(see page 173\)](#)
- [Debug Your DB2 Stored Procedures \(see page 174\)](#)
- [Debug Your Online IMS Programs \(see page 174\)](#)
- [Use Batch Link Schedule to Enhance your DB2 SP or IMS/DC Debugging Experience \(see page 174\)](#)
- [Import and Export Schedule Files \(see page 176\)](#)

This section describes additional considerations for using batch link in special environments.

DB2 Considerations

Use batch link to debug DB2 applications, provided that the following JCL conversion rules are implemented:

- The main program name (IKJEFT01) is not replaced by CAMRBL01 in the PGM= parameter of the // EXEC statement. If you are using the Call Attach Facility of DB2, your main program name is the application name. There are no special JCL conversion rules for this environment.
- SYSTSIN is modified to identify CAMRBL01 as the main program to be invoked by DB2.
- The EXEC option in INT1OPTS must identify the name of the application's main program, the name that is replaced by CAMRBL01 in the SYSTSIN file.
- If a LIBRARY parameter exists in the SYSTSIN file, remove it. You must add to the STEPLIB or JOBLIB concatenation the library containing the application's main program and the library containing the batch link executables.

Convert DB2 JCL using the Batch Link JCL Conversion dialog.

DB2 JCL Example

The following example shows a DB2 job step that has been converted for batch link:

```
//RUN      EXEC  PGM=IKJEFT01,COND=(4,LT),DYNAMNBR=20
//STEPLIB  DD  DISP=SHR,DSN=CAI.CAVHLOAD
//         DD  DISP=SHR,DSN=USER01.LOADLIB
//         DD  DISP=SHR,DSN=SYS2.DB2510.MAINT.SDSNLOAD
//SYSTSPRT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSTEM   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN     DD  *
/*
//SYSTSIN   DD  *
DSN  SYSTEM(D510)
RUN  PROGRAM(CAMRBL01) PLAN(DB2DEMO)
     PARM('ABEND0C7/TRAP(OFF)')
END
/*
//INT1OPTS DD  *
EXEC=DB2DEMO,PROFILE=USER01
/*
//INT1PARM DD  DISP=SHR,DSN=CAI.CAVHSAMP
```

```
//INT1LOAD DD DISP=SHR,DSN=CAI.CAVHLOAD
//INT1PNLL DD DISP=SHR,DSN=CAI.CAVHPNL1
//INT1MSG0 DD DISP=SHR,DSN=CAI.CAVHMSG0
//INT1PROF DD DISP=SHR,DSN=CAI.PROFLIB
//INT1CLIB DD DISP=SHR,DSN=CAI.CAVHSRC
//INT1CLOG DD SYSOUT=*
//INT1REPT DD SYSOUT=*
```

Debug Your DB2 Stored Procedures

Use batch link to debug DB2 stored procedures. Confirm with your systems programmer that the necessary changes (<https://docops.ca.com/display/CAITSD11/Batch+Interfaces+and+Compatibility#BatchInterfacesandCompatibility-CustomizeforDB2SPandIMS/DCDebugging>) have been made to the WLM environment to support debugging DB2 stored procedures. Also, confirm that if the scheduling feature is enabled, described in this section, that it is set up to debug your stored procedure. For more information on the required changes to the WLM environment, see [Batch Interfaces and Compatibility](https://docops.ca.com/display/CAITSD11/Batch+Interfaces+and+Compatibility#BatchInterfacesandCompatibility-CustomizeforDB2SPandIMS/DCDebugging). (<https://docops.ca.com/display/CAITSD11/Batch+Interfaces+and+Compatibility#BatchInterfacesandCompatibility-CustomizeforDB2SPandIMS/DCDebugging>)

DB2 SP SCHEDULING CONSIDERATIONS

Since DB2 SP currently runs under the WLM started task, the selection request for DB2 SP is made without the REGION/JOBNAME attribute. Thus REGION/JOBNAME Selection Phase is bypassed. You can continue to specify a REGION/JOBNAME field for DB2 SP schedule entries or you can leave that field blank.

Debug Your Online IMS Programs

Use batch link to debug your online IMS applications without the use of BTS. Confirm with your systems programmer that the DC transaction that you want to debug has its DC region set up for batch link testing. Also, confirm that if the scheduling feature, described in this section, is enabled, it is set up to debug your IMS transaction.

Use Batch Link Schedule to Enhance your DB2 SP or IMS/DC Debugging Experience

The batch link schedule feature helps you leverage your testing environment by letting you specify exactly which stored procedures or IMS/DC transactions you want the application to monitor. All other programs run unmonitored. This lets you use the same region for debugging as well as executing normal DB2 SP or IMS/DC workload.

Confirm with your systems programmer that the product has been initialized with this feature enabled. You can also confirm this by going into the product and choosing option 5, Batch Link. From there, choose option 4, Scheduling. From there, choose Option 1, supply a database type, and press Enter. If a short message of 'SCHEDULE TABLE NOT ANCHORED' displays, this feature has not been enabled. In this case, all DB2 SPs and IMS/DC transactions will run unmonitored.

Select DB2 SPs or IMS/DCs for monitoring

Entries in the schedule are organized top down and are tracked and managed by slot numbers (slot #1 through slot #nnnn). Entries are added using the first available scheme. Thus when an entry in the low slot number range (for example #2), is deleted, the next add is added to #2, using the first available scheme:

Slot Number	Region/Job Name	Program Name	TYPE	One Time
1 (top)	MON	SP*	DB2	Y
2	MON*	SP*	IMS	Y
3		SP*	DB2	Y
4	MONTH*	SP*	DB2	N
60 (bottom)				

Selection of an entry for monitoring is on a **first-and-best** match basis and in a top down fashion (slot #1 through slot #60). Select an entry for monitoring only after it has passed all the Selection Phases.

1. Enter Selection Phase - Match entries first against TYPE. Proceed to next phase if there is a match, otherwise examine the next entry.
2. REGION/JOBNAME Selection Phase - next REGION/JOBNAME of the entry having a matching TYPE is compared. If REGION/JOBNAME starts with an asterisk (*) or contains all blanks, any region or job name selection request would be treated as a match to the REGION/JOBNAME. Trailing blanks are treated as part of the name, thus the entire length (8 bytes) of REGION /JOBNAME is used for comparison (MON). Trailing asterisk makes the name generic, thus the characters and length to the left of the asterisk are used for comparison (MON). Proceed to the next phase if there is a match, otherwise examine the next entry.
3. Program Name Selection Phase - When the system finds a matching REGION/JOBNAME entry, it now matches its Program Name against the selection request. Trailing blanks are treated as part of the program name, thus the 8 byte Program Name including blanks is used for comparison. A trailing asterisk makes the program name generic, thus the characters and length to the left of the asterisk are used for comparison. When the program name matches that of the selection request, the program is monitored.

We will use the table (on the previous page) to illustrate monitor selection logic and flow. Assuming we have two batch link jobs:

1. JOBNAME = MONDAY; PROGRAM NAME = SPECIAL1; TYPE = DB2
2. JOBNAME = MONTH3; PROGRAM NAME = SPECIAL; TYPE = DB2

Job 1 is ready to be monitored. Attributes in the first batch link job are used as selection criteria:

1. Slot #1 Type Selection Phase ok. Match on TYPE DB2
2. Slot #1 REGION/JOBNAME Selection failed. MON (with 5 trailing blanks) does not match MONTHLY.
3. Slot #2 Type Selection Phase failed. No match on TYPE of DB2.
4. Slot #3 Type Selection Phase ok. Match on TYPE DB2
5. Slot #3 REGION/JOBNAME Selection ok. Match on REGION/JOBNAME (all blanks treated as a match)

6. Slot #3 Program Name Selection Phase ok. SP of SP* matches SP of SPECIAL1.
7. Slot #3 is deleted (ONE TIME set to Y)
8. PROGRAM SPECIAL1 is selected for monitoring.

Job 2 is now ready to be monitored. Attributes in the second batch link job are used as selection criteria:

1. Slot #1 Type Selection Phase ok. Match on DB2
2. Slot #1 REGION/JOBNAME Selection failed. MON (with 5 trailing blanks) does not match MONTHLY.
3. Slot #2 Type Selection Phase failed. No match on DB2.
4. Slot #3 deleted and is skipped.
5. Slot #4 Type Selection Phase ok. Match on TYPE DB2.
6. Slot #4 REGION/JOBNAME Selection Phase ok. Generic name MON of MONTH* matches MON of MONTH3
7. Slot #4 Program Name Selection Phase ok. Generic SP of SP* matches SP of SPECIAL.
8. Slot #4 is deleted
9. Slot #4 is selected for monitoring.

Assuming Job 2 is submitted before Job 1, Slot #3 is again selected for monitoring. However because its One Time Schedule is set to Y, Slot #3 is deleted before Job 1's selection request is made. In this case, Job 1 runs unmonitored, since MONTH of MONTH* (slot #4) does not match MONDAY (Job 1's JOBNAME).

Import and Export Schedule Files

Importing a schedule provides you with an easy way to maintain a persistent schedule across IPLs. You can also make significant scheduling changes instead of changing individual schedules manually using the online change option.

You can create the schedule import file manually when you first use the scheduling feature, or you can create it later using the schedule export option, since you can activate the scheduling feature with an empty schedule and later add individual schedules (using the online Schedule Add option) as the need arises.

The schedule export option is an easy way to create a backup copy of your fine-tuned schedule or a uniquely tailored application system schedule to facilitate systems testing.

Special Processing for DB2/IMS Schedule Export and Import

If a DB2/IMS schedule import file has the JOBNAME or region name specified as * (wildcard for entire name), that line is not imported. A * in column 1 was considered as a comment on import. This solution changes the * for jobname to ? on export and then on import changes the ? back to a *.

Sample Schedule Import File

The following example shows a sample import file and its supported record layout:

```
*****
* CA BATCH LINK SCHEDULES SAMPLE IMPORT
* ASTERISK IN COLUMN 1 DENOTES A COMMENT LINE.
* THE SYSTEM WILL VALIDATE DATABASE TYPE FIELD, AND ONE TIME SCHEDULE FI ELD.
* VALID ONES ARE: DB2 OR IMS FOR DATABASE TYPE; Y OR N FOR ONE TIME SCHEDULE.
* OTHER FIELDS ARE ACCEPTED ASIS, WITHOUT VALIDATION.
* FORMAT OF IMPORT/EXPORT RECORD:
* COLUMN 1 - 8 REGION/JOBNAME (8 BYTES)
* COLUMN 10 - 17 PROGRAM NAME (8 BYTES)
* COLUMN 19 - 19 ONE TIME SCHEDULE (1 BYTE. Y OR N)
* COLUMN 21 - 23 DATA BASE TYPE (3 BYTES. DB2 OR IMS)
* ---+---1---+---2---+---3---+---4---+---5---+---6---+---7--
BIMNTH01 GL001 Y DB2
BIMNTH01 ASSET* Y DB2
BIMNTH01 SALES* Y DB2
WKLYPAY TIMEATTN Y DB2
QTRLY080 OXLYSABN Y IMS
QTRLY090 SAP001 Y IMS
QTRLY100 SAP002 Y IMS
```

Sample Schedule Export File

The following example shows a sample export file with a system generated record layout:

```
*****
* CA BATCH LINK SCHEDULES SAMPLE EXPORTED FROM SID: CA31 ON 07/14/2014 AT 09:41
* ASTERISK IN COLUMN 1 DENOTES A COMMENT LINE.
* THE SYSTEM WILL VALIDATE DATABASE TYPE FIELD, AND ONE TIME SCHEDULE FI ELD.
* VALID ONES ARE: DB2 OR IMS FOR DATABASE TYPE; Y OR N FOR ONE TIME SCHEDULE.
* OTHER FIELDS ARE ACCEPTED ASIS, WITHOUT VALIDATION.
* FORMAT OF IMPORT/EXPORT RECORD:
* COLUMN 1 - 8 REGION/JOBNAME (8 BYTES)
* COLUMN 10 - 17 PROGRAM NAME (8 BYTES)
* COLUMN 19 - 19 ONE TIME SCHEDULE (1 BYTE. Y OR N)
* COLUMN 21 - 23 DATA BASE TYPE (3 BYTES. DB2 OR IMS)
* ---+---1---+---2---+---3---+---4---+---5---+---6---+---7--
BIMNTH01 GL001 Y DB2
BIMNTH01 ASSET* Y DB2
BIMNTH01 SALES* Y DB2
WKLYPAY TIMEATTN Y DB2
QTRLY080 OXLYSABN Y IMS
QTRLY090 SAP001 Y IMS
QTRLY100 SAP002 Y IMS
```

An asterisk is used as a wild card character to match all or part of a name. An asterisk in the first column of a name indicates that any name is matched. However, if any region/jobname will be matched, that is - the entire region/jobname is specified as a wildcard, then column 1 must be a ? in the import file. This will be changed to an * when imported. On an export, an * in column 1 for the wildcard of an entire region/jobname will be changed to a ?. When entering the schedule on the panel, use the * in column 1 to specify matching all region/jobnames.

CA InterTest Batch Utilities

This section describes the functionality provided in the Utilities option. Using this option, you can display and update your symbolic files (PROTSYM).

- [Display the Contents of the Symbolic File \(see page 178\)](#)

- [Add a Program to the PROTSYM File \(see page 178\)](#)
- [Exclude Programs from Auto-monitoring \(see page 179\)](#)
- [Work With Recorded Debug Sessions \(see page 179\)](#)

When you select option U of the Primary Option Menu, the Utilities Menu displays. This menu consists of three options - one for listing and displaying the contents of the symbolic file, one for adding listings to a symbolic file, and one for defining programs to be excluded from auto-monitoring.

Display the Contents of the Symbolic File

After selecting option 1 from the Utilities Menu, the PROTSYM Member Selection panel displays:

```
----- CA InterTest Batch PROTSYM Member Selection -----
COMMAND ==> SCROLL ==> CUR
-----
PROTSYM Dsname ==> 'CAI.PROTSYM'
-----
Cmd  Program  Date      Time      Size Language  Attributes
***** TOP OF DATA *****
      CAMRCOB2 12/03/2014 13:56:01   83 COBOL 05/390
      CAMRCOB2 12/03/2014 13:55:47  144 COBOL 05/390
      ITBMAIN  08/16/2014 08:46:02   16 COBOL II
      ITBSUB1  08/06/2014 18:14:59   10 COBOL II
***** BOTTOM OF DATA *****
```

This panel lists the contents of a given PROTSYM file. From this panel you can enter a valid PROTSYM file in the PROTSYM Dsname field. This displays the contents of this symbolic file. An example is shown in the previous screen. This list displays the program name, the date and time that the program was compiled, the number of records in the PROTSYM that this member is using, the language that was used, and any attributes that the member has.

Using an S line command displays the listing for that particular program.

Add a Program to the PROTSYM File

After selecting option 2 of the Utilities Menu, the PROTSYM Add Program Listings panel displays. Populate a PROTSYM file with one or more listings using this panel.

The PROTSYM Add Program Listings panel contains the following fields:

- **PROTSYM Dsname**
The name of the PROTSYM data set
- **Listing Dsname**
The name of the listing data set
- **Library Type**
The listing library data set type. Specify one of the following:
 - PDS - Partitioned Data Set
 - SEQ - Sequential Listing File

- PAN - CA-Panvalet Library
- LIB - CA-Librarian Library
- NDV - CA Endeavor SCM Library
- **From Member**
The starting member name for the listing data
- **To Member**
The ending member name for the listing data set
- **View Messages**
Identifies the message display options. Specify one of the following:
 - ALL - Display all messages
 - NONE - Display no messages
 - RC - View a single message for each member containing the highest return code

To process just one member, enter the member name in the From Member field. To process a range of members, enter the starting member in the From Member field and the ending member in the To Member. You can also use a prefix ending with an asterisk wildcard to process several members with a common prefix.

You can process a range of members using a batch utility that is described in the member IN25SYMD in the CAVHPROC data set.

Exclude Programs from Auto-monitoring

After selecting option 3 of the Utilities Menu, the Excluded Programs panel displays.

Use this panel to restrict auto-monitoring of programs when wildcarding is used on the Monitor Control panel. Use a single trailing asterisk to define a prefix, if desired. If a program name or wildcarded prefix has been excluded, it can only be monitored by explicitly defining the full name on the Monitor Control panel. To see the list of program names that are already excluded by default, type DEFAULTS on the command line.

Work With Recorded Debug Sessions

The Debug Session Display shows the contents of the debug session library file. Select Option 4 from the Utilities Menu to display this panel. You can also display this panel by using the MLOG command from a debugging session.

```

----- CA-InterTest Batch Debug session Selection -----
COMMAND ===>                                     SCROLL ===> CUR

Type B to browse session      D to delete session

```

```

-----
Cmd  Session  UserID   Jobname  Date      Time      Description
***** TOP OF DATA *****
@NAME      VUOLO01   VUOLO01  2016258  111412  MY DESCRIPTION
ASMINCL                                Not created by MLOG
ASM1       VUOLO01   VUOLO01  2016258  101424  WHY DOES LAURA MAKE ME DO THIS
ASSEM9     VUOLO01   VUOLO01  2016270  165101  ASSEMBLER TEST SESSION
BATCH      VUOLO01   VUOLO01D 2016257  162002  BATCH DEBUG SESSION
CAMRASM    VUOLO01   VUOLO01  2016258  111258  ASSEMBLER TEST
COBOL9     VUOLO01   VUOLO01  2016258  095712  TEST COBOL PROGRAM
DEMOCANC   VUOLO01   VUOLO01  2016292  104047  CANCEL DEMO TEST
DEMOINCL                                Not created by MLOG
DEMOSTOP   VUOLO01   VUOLO01  2016292  103849  STOP DEMO
HR          VUOLO01   VUOLO01  2016263  101008  HR NEW PROGRAM
LAURA1    VUOLO01   VUOLO01  2016263  100320  LAURA'S DEBUG SESSION
MYPROG     VUOLO01   VUOLO01  2016271  083223  MY TEST PROGRAM
PAYROLL    VUOLO01   VUOLO01  2016263  100728  PAYROLL DEBUGGING
PLIINCL                                Not created by MLOG
***** BOTTOM OF DATA *****

```

Use this panel to display the contents of your debug session (command library) file (INT1CLIB). This panel will display the session name, userid, jobname, date and time that the debug session was created by the MLOG command, as well as a description of the session. It will also display command library files that were created outside of the MLOG command.

This panel displays the following information:

- Session name
- Userid, jobname
- Date and time that the debug session was created
- Description of the session
- Command library files that were created outside of the MLOG command

The lines marked as '*ACTIVE' are sessions that are currently still in progress or being edited outside of InterTest Batch. You cannot delete these sessions from the display panel.

You can use the following line commands in the action field:

- **B**
Browse the command file member
- **D**
Delete the command file member, if it is not active
- **L**
Load the command file member into the current debug session

- **S**
Stop the active debug session
- **C**
Cancel the active debug session



You can only use the **L**, **S**, and **C** commands from an active debugging session.

You can use the following primary commands at the command line:

- **Find**
Locate a string on the panel. Use quotes if the text includes embedded spaces.
- **RFind**
Repeat the prior Find command
- **LOAD member**
Load the specified member into the current debug session
- **SELECT member**
Display the contents of the command file member
- **STOP member**
Stop the active debug session
- **CANCEL member**
Cancel the active debug session
- **DELETE member**
Delete the saved debug session, if it is not active

DDnames and Sample CLISTs

- [DDnames Required for Proper Execution \(see page 181\)](#)
- [Optional DDnames \(see page 182\)](#)
- [Sample CLISTs to Invoke the Application \(see page 183\)](#)
- [Other CLISTs \(see page 183\)](#)

DDnames Required for Proper Execution

This section contains a list of the ddnames that are used by the application and a brief description of their use.

DCB information, where given, is that which is set and used by the application.

DDname	Description
--------	-------------

INT1LOAD	Defines the partitioned data set (PDS) that contains the load modules. This library is built during installation.
INT1MSG	Defines the partitioned data set that contains the application messages. This library is built during installation.
INT1PARM	Defines the partitioned data set that contains the application parameters. This library is built during installation.



Note: The parameters are internal to the application and are documented. For information about how these parameters are documented, see [Customize Options in CA InterTest Batch Options \(https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options\)](https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options).

INT1PNLL	Defines the partitioned data set that contains the panel definitions. This library is built during installation.
INT1PROF	Defines the partitioned data set used to maintain user-related data from session to session. This library must be defined by the installation. For information about how to allocate the profile library, see Allocate PROFLIB in CA InterTest Batch Options (https://docops.ca.com/display/CAITSD11/CA+InterTest+Batch+Options) .

Optional DDnames

These ddnames are used during certain processes. If you are using the designated application facility, allocate these files before using that part of your product.

It is prudent to include the necessary allocation of ddnames to DSN in the CLIST that invokes the application.

DDname	Description
INT1PRNT *	Defines a sequential file that is used to print the screen images. If it is not allocated, the screen image feature is disabled. For more information about printing screens, see the print key description in Program Function and Program Access Keys in Panels and Functions (see page 16) .



Note: This DD is used only under CA Roscoe.

LRECL=121, BLKSIZE=6171, RECFM=FBA

INT1REPT	Defines a sequential file that is used to store the HISTogram and XSUM reports. If it is not allocated, the HIST and XSUM commands are disabled. For more information about the reports, see the HIST and XSUM Report Command descriptions in Debugging Commands (see page 40) .
----------	--

LRECL=131, BLKSIZE=6157, RECFM=FBA

INT1ALIB

DDname	Description
	Defines the partitioned data set that contains allocation sets that are saved under option 3, ALLOCATION. If it is not allocated, the save and retrieve features are disabled. For more information about allocating data sets and saving the allocations, see Allocations Facility for ISPF (see page 136).
	LRECL=150, BLKSIZE=3150, RECFM=FB
INT1CLIB	Defines the partitioned data set that contains application commands to be used by the INCLUDE command. If it is not allocated, the INCLUDE command is disabled. For more information, see the Include Control Command in Debugging Commands (see page 40).
	LRECL=80, RECFM=FB
INT1CLOG	Defines a sequential file that is used for the application session log. If it is not allocated, logging is disabled. For more information about the session log, see Session Log Facility (Review Debugging Session) in Debugging Commands (see page 40).
	LRECL=80, BLKSIZE=6160, RECFM=FB
SYSOUT	Although SYSOUT is not used by the application, it is recommended that it be allocated to the terminal to prevent ABENDS while processing COBOL DISPLAY or EXHIBIT verbs that may have been left in the program.

Sample CLISTs to Invoke the Application

CAI.CAVHCLS0 contains five sample CLISTs for invoking the application in different program testing environments. The following table describes the sample CLISTs:

Name	Description
CAMRCLST	Sample CLIST for application testing.
CAMRCIMS	Sample CLIST for testing of IMS programs in foreground.
CAMRCBTS	Sample CLIST for testing of IMS programs using BTS input in foreground.
CAMRCDB2	Sample CLIST for testing of DB2 programs in foreground.
CAMRDRVR	Sample CLIST that provides a front-end to the other four CLISTs.

Other CLISTs

The application contains several other CLISTs for your use. They are described in the following table:

Name	Description
CAMRSTRT	This gets called under an ISPF NEWAPPL and calls the program to start the application. CAMRKEYS can be called from this CLIST to set the PF Keys
CAMRKEYS	This gets called from the CLIST CAMRSTRT to set up initial PF key settings for this APPL. By default, all of the PF keys are set to an initial value. By default, the PF keys are only set once. To have the PF keys reset, modify the variable CHANGED.
INT1ALIB	This CLIST conditionally creates an ALIB data set. It can be called from your startup CLIST.

INT1CLIB	This CLIST conditionally creates a command library data set. It can be called from your startup CLIST.
INT1CLOG	This CLIST conditionally creates a session log data set. It can be called from your startup CLIST.
INT1PROF	This CLIST conditionally creates an application profile data set. It can be called from your startup CLIST.
INT1REPT	This CLIST conditionally creates a user report data set. It can be called from your startup CLIST.

Batch Abend Analysis

CA SymDump Batch provides comprehensive abend diagnostic information for batch abends in test and production environments. Its formatted reports make it easy for programmers to resolve abends without having to decipher system dumps. It supports all languages, with special symbolic support for COBOL, PL/I, and Assembler. It also includes detailed database information for abending IMS, DB2, and CA IDMS/DB applications. After installed, CA SymDump Batch is automatically available to every user.

Components

The CA SymDump Batch solution consists of the following components:

- **Abend Handler/Reporter**
Traps application abends and saves report data in the central VSAM repository.
- **CAIPRINT Formatter**
Generates formatted reports from the report data.
- **Viewing Abend Reports**
Reports can be viewed using:
 - CAIPRINT Repository Viewer in TSO/ISPF
 - Eclipse-based Graphical User Interface



Note: For more information on the Eclipse Interface see [Release Information](https://docops.ca.com/display/CAITSD11/Release+Information) (<https://docops.ca.com/display/CAITSD11/Release+Information>) and [Installing](https://docops.ca.com/display/CAITSD11/Installing) (<https://docops.ca.com/display/CAITSD11/Installing>).

- **Symbolic Postprocessors**
Writes symbolic information extracted from COBOL, PL/I, or Assembler program listings to the PROTSYM file.
- **Utilities**
Lets programmers perform the following tasks:
 - Provide load library reports that identify and describe COBOL CSECTs in detail
 - Generate a report describing which CA SymDump Batch modules are currently loaded in the CSA area of LPA
 - Generate a report detailing the options set as installation defaults
 - List the contents of the symbolic files

- List and maintain the contents of the central VSAM repository
- Display Dynamic Symbolic Support (DSS) options for local ISPF session in a user's personal ISPF profile.

Using Symbolic Support

CA SymDump Batch reports on all abending programs and provides additional information for COBOL, PL/I, and assembler programs. For each of these programming languages, symbolic support can be used to enhance the content of your Abend or Snap reports. Symbolic support is not required when using this product.

This section provides a brief overview of symbolic support. A complete description of symbolic support, including all of the symbolic postprocessors and utilities, can be found in [Symbolic Support](https://docops.ca.com/display/CAITSD11/Symbolic+Support) (<https://docops.ca.com/display/CAITSD11/Symbolic+Support>) .

Symbolic Support Features

When symbolic information is available for your programs, the following additional information can be found in your Abend or Snap reports:

- COBOL
 - Abending or last executed COBOL source statement
 - Referenced variables on the last executed statement
 - Merged variable displays
- PL/I
 - Abending or last executed PL/I source statement
 - Referenced variables on the last executed statement
 - Merged variable displays
- Assembler
 - Abending or last executed Assembler source statement
 - Merged storage displays for active USINGs

For more information about how post-processors store symbolic information, see [Symbolic Support](https://docops.ca.com/display/CAITSD11/Symbolic+Support) (<https://docops.ca.com/display/CAITSD11/Symbolic+Support>) .

Use Existing CSL Files

CSL files created by CA Optimizer, CA Optimizer/II, or by an earlier release of CA SymDump Batch can also be used as a source of symbolic information.

There is no need to convert your existing CSL files into the PROTSYM format.

However, while CSL files can still be used as input for report generation, CA SymDump Batch will no longer create new CSL members.

Access Symbolic Information at Execution Time

After the symbolic information is stored in a PROTSYM or CSL file, the information can be accessed by CA SymDump Batch to provide symbolic support for your Abend and Snap reports.

When an Abend or Snap report is initially written to CAIPRINT at execution time, CA SymDump Batch automatically searches each of the PROTSYM and CSL files defined in your installation defaults member, CAOUDFRX. You can define as many symbolic files as needed for your installation, and you can freely mix PROTSYM and CSL files. You can display the list of symbolic files defined at your installation using the SYM command from the CAIPRINT Repository Viewer.

You can also add one additional symbolic file, a PROTSYM or CSL, to your execution JCL using the CAISYM DD statement. Your installation may use an alternate DD name for CAISYM. Check with your system administrator if you are not sure. Alternatively, you can have one symbolic file dynamically allocated at execution time using the SYMDSN option in the CAIOPTS file.

The search order for symbolic information at execution time is as follows:

- The symbolic file defined using the CAISYM DD, if any
- The symbolic file defined using the SYMDSN option in CAIOPTS, if any
- The globally defined symbolic files from your installation defaults member CAOUDFRX at the time of execution, in the order that they are defined, if any

If no exact match is found, the symbolic information retrieved from the symbolic file defined by the CAISYM DD or by the SYMDSN option is used. If neither is available, the most recent symbolic information is selected.



Note: A symbolic mismatch can cause unpredictable results or misleading symbolic information in the formatted reports. This situation can be mitigated by installing and activating DSS. For more information, see [Dynamic Symbolic Support \(see page 320\)](#).

Add Symbolic Information Using the Viewer

Symbolic information does not need to be available at the time an Abend or Snap report is created, provided that your reports are being written to a CAIPRINT repository data set. If so, you can easily add symbolic information after the report is written to the repository, when you actually view the report. Simply load the COBOL, PL/I, or Assembler listing file into the PROTSYM using the appropriate postprocessor, then select the report for viewing.

You can add the symbolic information to any of the symbolic files defined in your installation defaults member, CAOUDFRX. Alternatively, you can add the symbolic information to your own personal symbolic file and then define that symbolic file to the CAIPRINT Repository Viewer as a *supplemental* symbolic file. Each user can define as many as sixteen supplemental PROTSYM or CSL files that are only searched when that individual user selects a report from the repository for viewing. For more information about supplemental symbolic files, see [Using the CAIPRINT Repository Viewer \(see page 263\)](#).

When you select a report from the repository for viewing, CA SymDump Batch tries to find an exact match for COBOL and PL/I programs using the date/time stamps in both the listing and the object code.

The search order for symbolic information when using the viewer is as follows:

- The symbolic file that was defined using the CAISYM DD when the program was executed, if any
- The symbolic file that was defined using the SYMDSN option in CAIOPTS when the program was executed, if any
- The supplemental symbolic files that are defined for your userid at the time the report is selected for viewing, in the order that they are defined, if any
- The globally defined symbolic files that are defined in your installation defaults member CAOUDFRX at the time the report is selected for viewing, in the order that they are defined, if any

If no exact match is found, the symbolic information retrieved from the symbolic file defined by the CAISYM DD or by the SYMDSN option is used. If neither is available, the most recent symbolic information is selected.



Note: A symbolic mismatch can cause unpredictable results or misleading symbolic information in the formatted reports. This situation can be mitigated by installing and activating DSS. For more information, see [Dynamic Symbolic Support \(see page 320\)](#).

Reporting

CA SymDump Batch provides reports when a system dump is requested as the result of an ABEND, or when a snap request is made using a call to CAODSNAP. Until that time, there is no intervention of any kind. For more information, see Abend Reporting with Language Environment (LE) later in this section.

CA SymDump Batch reports on all programs, with additional information provided for COBOL, PL/I, and Assembler programs. There are several levels of information that may be available depending on the programming language and the existence of symbolic information.

The contents of a report are completely controlled by the installation defaults and override options. It is also possible to receive a system dump in addition to abend reports. Exclusion criteria can be selected at installation time to allow for system dumps instead of abend reports for certain ABENDs.

CA SymDump Batch generates two different reports:

- The abend report eliminates much of the searching and guesswork involved in debugging programs. The abend report presents diagnostic information and key storage data information in a straightforward format.
- The Snap report provides snapshots of your program's data areas and other pertinent program information. To invoke the Snap report, place a CALL at the spot in your program where you want the snapshot to be taken. You can specify multiple snaps in a single program execution.

Hierarchy of Information

When CA SymDump Batch produces an abend report, all of the programs that are active at the time of the ABEND are included, unless the ACTONLY option is set to ON, or unless individual programs are excluded from reporting in your installation defaults.

Several factors affect how much information is provided for each program, including:

- For COBOL or PL/I programs, storage areas and control blocks are broken out and formatted, and compiler statistics are provided.
- If your PROTSYM (or CSL) contains symbolic information for an active COBOL, PL/I, or Assembler program, the reports include the last executed source statement, the names and values of the variables, and more.

Usage Considerations

When using abend reporting, be aware of these considerations:

- CA SymDump Batch does not provide any reporting for programs that abend under CICS or other teleprocessing environments.

- CA SymDump Batch is only supported on z/OS systems.
- For the most productive use of CA SymDump Batch, your applications should use standard IBM-type program linkage conventions whenever possible.
- CA SymDump Batch does not provide any reporting if the abending program resides above the bar.

Reporting Options

You can change the default value of most CA SymDump Batch options by coding an override in the options file at execution time. The default DDNAME of the options file is CAIOPTS, which you can change at installation time. At execution time, you can allocate the options file dynamically, allowing each user to customize their options differently without coding any additional JCL.

- [Standard Options \(see page 198\)](#)
- [Formatting Options \(see page 198\)](#)
- [VSAM Control Block Options \(see page 198\)](#)
- [Data Management Control Block Options \(see page 199\)](#)
- [Coding Options \(see page 199\)](#)
- [Override Defaults \(see page 199\)](#)

The following list contains all reporting options:

- **ACB ON|OFF**
Prints access method control blocks for open VSAM files in an Abend or Snap report.
Synonym: None
Component: Reporting
When used: Execution time
- **AMB ON|OFF**
Prints access method blocks for open VSAM files in Abend or Snap Reports.
Synonym: None
Component: Reporting
When used: Execution time
- **AMBL ON|OFF**
Prints an access method block list for open VSAM files in an Abend or Snap report.
Synonym: None
Component: Reporting
When used: Execution time
- **ASMINST ON|OFF**
Displays the abending assembler instruction and associated operands on the Abend page.
Synonym: None
Component: CAIPRINT Formatter
When used: Execution time, view time
- **BINFRMT DEC|HEX**
Specifies whether to display binary data values in decimal display or hexadecimal format on the merged Data Division displays of an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **BLLMAX 512|nnnn**

Because the length of an 01-item in the linkage section cannot always be determined, this option specifies the maximum number of bytes which are formatted for a BLL cell whose range is unknown. The maximum value is 4096.

Synonym: None

Component: Reporting

When used: Execution time

▪ **BLOCKS ON|OFF**

A data management control block option which sets all control block options, including all VSAM control block options, either ON or OFF for an Abend or Snap report. This option overrides any previous control block options.

Synonym: None

Component: Reporting

When used: Execution time

▪ **DB2ACTIV ON|OFF**

Indicates whether to display summary information for only those packages and DBRMs that are active at the time of an Abend or Snap. If the option is set to OFF, summary information is displayed for all of the packages and DBRMs defined to the DB2 plan.

Synonym: None

Component: Reporting

When used: Execution time

▪ **DCB ON|OFF**

Prints data control blocks for all open files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

▪ **DEB ON|OFF**

Prints data extent blocks for all open files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

▪ **DUMP ON|OFF**

Specifies whether to force or suppress a system dump after it has completed its processing.

Synonym: None

Component: Reporting

When used: Execution time

▪ **FILES ON|OFF**

Specifies whether to display all open files not already shown in the File Section display of an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ FIRST128 ON|OFF

Controls whether to print only the first 128 bytes of the current record of each open file or the entire record. This option affects both the File Section display and Open Files report of an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ GRPADDR ON|OFF

Controls whether to display the base locator, displacement and address of a group item on the merged Data Division displays of an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ IOB ON|OFF

Prints input/output blocks for all open files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

▪ LINECOUNT 60|nnn

Specifies the number of lines on each page of output at execution time or view time. where nnn can be any integer from 10 to 255, inclusive.

Synonym: LINECNT

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ LINKAGE ALL|NONE

Specifies whether to display (ALL) or suppress (NONE) the program's Linkage Section in an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ LISTLINE=0|nn

Specifies the number of additional listing lines to be merged into a report before and after the source statement at abend, snap, or transfer.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ LOCALSTOR ALL|NONE

Specifies whether to display (ALL) or suppress (NONE) the program's Local-Storage Section in an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ LOGROS ON|OFF

Specifies whether to send a message describing the abend to the CA Roscoe user who submitted the job. You must establish a list of CA Roscoe job names in the CA SymDump BatchCA SymDump Batch defaults using the ROSCOE parameter.

Synonym: None

Component: Reporting

When used: Execution time

▪ **LOGTSO ON|OFF**

Indicates whether to send a message describing the abend to the TSO user who submitted the job.

Synonym: None

Component: Reporting

When used: Execution time

▪ **LOGUNI ON|OFF**

Indicates whether to send a message describing the abend to the CA Unicenter NSM console.

Synonym: None

Component: Reporting

When used: Execution time

▪ **MEMMAP ON|OFF**

Produces a display of the memory map (TGT, PGT, and DSA) in an Abend or Snap report.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **MERGEDB ON|OFF**

Indicates whether to symbolically map DB2, IDMS, and IMS data. If the option is set to ON, data item information such as the name, picture clause, and usage type is merged with the dump output for those areas.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **MRGAUTO ON|OFF**

Indicates whether to symbolically map the data for PL/I automatic storage. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **MRGBASED ON|OFF**

Indicates whether to symbolically map the data for PL/I based variables. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **MRGCNTLD ON|OFF**

Indicates whether to symbolically map the data for PL/I controlled variables. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGDATA ON|OFF

Indicates whether to set all data merging options either ON or OFF for an Abend or Snap report. This option overrides any previous merging options.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time

▪ MRGDSECT ON|OFF

Indicates whether to symbolically map the data for Assembler storage areas when a USING is active. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGFILES ON|OFF

Indicates whether to symbolically map the data in the File Section display of an Abend or Snap report. If the option is set to ON, data item information such as the name, picture clause, and usage type is merged with the data for the current logical record.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGLINK ON|OFF

Indicates whether to symbolically map the data in the Linkage Section display of an Abend or Snap report. If the option is set to ON, data item information such as the name, picture clause, and usage type is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGLOCAL ON|OFF

Indicates whether to symbolically map the data in the Local-Storage Section display of an Abend or Snap Report. If this option is ON, data item information such as the name, picture clause, and usage type is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGPARDS ON|OFF

Indicates whether to symbolically map the data for PL/I parameter storage. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ MRGSTAT ON|OFF

Indicates whether to symbolically map the data for PL/I static storage. If this option is ON, data item information such as name and definition is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **MRGWORK ON|OFF**

Indicates whether to symbolically map the data in the Working-Storage Section of an Abend or Snap report. If the option is set to ON, data item information such as the name, picture clause, and usage type is merged with the data.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **NDVRASM=OFF|ON**

Indicates whether the symbolic file should always be dynamically populated for assembler programs which are not LE enabled. This option will only take effect when dynamic symbolic support (NDVRDSS) is active for CA Endevor SCM.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **NDVRDSN=dsname**

Specifies the name of the symbolic file which will be dynamically populated when dynamic symbolic support (NDVRVSS) is active for CA Endevor SCM. The data set specified must be a PROTSYM file. When specified, it will automatically be included in the symbolic file search list.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **NDVRDSS=OFF|ON**

Indicates whether dynamic symbolic support is active for CA Endevor SCM. This option will not take effect at view time, unless it was also specified at execution time.

Synonym: None

Component: Reporting, CAIPRINT Formatter

When used: Execution time, view time

- **OCCURS 1|nnnnnnnn|MAX**

Specifies whether to symbolically map the maximum number of table occurrences when using the merging options. If the option is set to a value lower than the number of occurrences in the table, the remainder of the table occurrences are dumped in hexadecimal format. If OCCURS MAX is specified, the option is set to its maximum value of 16777215.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **PLH ON|OFF**

Prints place holders for open VSAM files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

- **PRTREPT BOTH|REPOS|PRTDD**

Indicates whether to write the CAIPRINT report to the central VSAM repository (REPOS), the CAIPRINT DD (PRTDD), or both.

Synonym: None

Component: Reporting

When used: Execution time

- **READLL=ON|OFF**

Indicates whether CA SymDump Batch will attempt to read LINKLIST libraries to obtain module information for Abend and Snap reports. If READLL is OFF, CA SymDump Batch may not be able to report on those programs in the application that were loaded from LINKLIST.

Synonym: None

Component: Reporting

When used: Execution time

- **REGMAX (xxx,yyy)|(128,256)**

Indicates how many bytes of addressable storage to display before and after the address in each register for an Assembler program. xxx represents the number of bytes before and must be within the range 0-32767. yyy represents the number of bytes after and must be within the range 0-4096. The REGMAX values must be enclosed in parentheses, and if both xxx and yyy are specified, they must be separated by a comma. If only xxx is specified, yyy is assumed to have the same value.

Synonym: None

Component: Reporting

When used: Execution time

- **RPL ON|OFF**

Prints a Request Parameter List for open VSAM files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

- **RPTSZ80 ON|OFF**

Indicates whether to force all data in dump format to appear in an 80-column format.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **SAVEAREA ON|OFF**

Produces the Save Area Trace on an Abend report. The Save Area Trace displays the chained register save areas starting at the abending module.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **SAVEHEAP ON|OFF**

Indicates whether to save the LE heap storage when a report is created. SAVEHEAP ON is required at execution time to use SHOWHEAP ON for LE enabled COBOL or Assembler programs.

Synonym: None

Component: Reporting

When used: Execution time

- **SHOWHEAP ON|OFF**

Indicates whether to display the LE or PL/I heap storage when a report is displayed. SAVEHEAP ON is required at execution time in order to use SHOWHEAP ON for LE enabled COBOL or Assembler programs.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **SHOWUNMRG ALL|NONE|TBLS**

Indicates whether to display storage for unmerged data when symbolic merging has been requested for the section. If the section is being displayed in the standard dump format, SHOWUNMRG has no effect. Specify ALL to always display unmerged data, NONE to never display unmerged data, or TBLS to display unmerged data only for table elements after the OCCURS limit has been reached for each table.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

▪ **SNAP 0|nnnnn|ON|OFF**

Indicates whether to generate a Snap report. You must specify the number of SNAPS to allow using nnnnn. The maximum number allowed is 32,767. To use SNAP, you must include a call to CAODSNAP in your source code wherever you want a snapshot to be taken. You may use the SNAP feature in the same run with the INTERCEPTS feature. If your program contains SNAP calls and you want to run it without them, you do not need to recompile; simply do not specify the SNAP option at all. **WARNING!** Each call to CAODSNAP generates a separate SNAP report in your central VSAM repository. If your program contains multiple SNAP calls (or a SNAP call within a loop), you can write the output to the CAIPRINT DD alone, to avoid flooding the repository. [See the PRTREPT option.](#)

Synonym: None

Component: Reporting

When used: Execution time

▪ **SYMDSN dsname**

Specifies to dynamically allocate the data set name of a symbolic file for retrieving symbolic information. The data set can be a PROTSYM file or a CSL library. The DDNAME used for dynamic allocation is determined by the value of SYMDD in the installation defaults module CAOUDFRX. SYMDSN is ignored when that DDNAME is already allocated. Using SYMDSN does not prevent the searching of other symbolic files defined in your installation defaults.

Synonym: None

Component: Reporting

When used: Execution time

▪ **UCB ON|OFF**

Prints unit control blocks for all open files in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

▪ **USERID 'your text'| 'CA'**

Indicates a user identification statement (1-60 bytes). You can change the default title during product installation. It is included in all report headings.

Synonym: None

Component: Reporting

When used: Execution time

▪ **VSAMCAT=OFF|ON**

Indicates whether to format the VSAM catalog information for each open VSAM file.

Synonym: None

Component: Reporting

When used: Execution time

- **VSAMIDX ON|OFF**

Prints the VSAM indexed control blocks for the index portion of an open VSAM file along with the control blocks for the data portion in an Abend or Snap report.

Synonym: None

Component: Reporting

When used: Execution time

- **WORKSTOR ALL|NONE**

Indicates whether to display (ALL) or suppresses (NONE) the program's Working-Storage Section display.

Synonym: None

Component: CAIPRINT Formatter

When used: Execution time, view time

- **WTO ON|OFF**

Indicates whether to send a message describing the abend to the job log (using WTO with ROUTCODE=11).

Synonym: None

Component: Reporting

When used: Execution time

Standard Options

Standard options are specified using the CAIOPTS data set.

For more information about application examples, see [Reporting \(see page 189\)](#).

Formatting Options

The formatting options can be modified at execution time or view time. The execution-time options are specified using the CAIOPTS data set. The view-time options are specified through the Report Options panel in the CAIPRINT Repository Viewer. For more information about how to override these options at view time, see [CAIPRINT Repository Viewer \(see page 263\)](#).

For more information about application examples, see [Reporting \(see page 189\)](#).

VSAM Control Block Options

VSAM Control Block options are specified using the CAIOPTS data set. These options control which VSAM Control Blocks are displayed on the Open Files report: ACB, AMB, AMBL, PLH, RPL, and VSAMIDX. The Data Management Control Blocks, including VSAM Control Blocks, for all open files in the job step optionally display.



Note: BLOCKS ON or OFF turns all VSAM control block options ON or OFF. For example, to display open ACBs and no other I/O control blocks specify the following:

BLOCKS OFF ACB ON.

No Data Management Control Blocks, except open ACBs, are displayed.

Data Management Control Block Options

Data Management Control Block options are specified using the CAIOPTS data set. The Data Management Control Blocks, including VSAM Control Blocks, optionally display on the File Section Display. These options control which I/O blocks are displayed on the Open Files Report: BLOCKS, DCB, DEB, IOB, and UCB.



Note: To print all control blocks except one, specify BLOCKS ON with the block you do not want printed specified as OFF. For example, to display all Data Management Control Blocks, except UCBs, specify:

```
BLOCKS ON
UCB OFF
```

Coding Options

The general rules for coding keywords and their options at execution time and view time are as follows:

- Use one or more spaces as delimiters for options.
- Use an equal sign (=) to separate a keyword from its option value. For example, DUMP=ON and DUMP ON are both acceptable.
- Use positions 1 through 72 and ignore positions 73 through 80.

You can specify the following while coding keywords:

- Options in any order; if you repeat an option, the last specified occurrence is applied.
- Any number of options on a line; however, do not split options between two lines.

You can also do the following:

- Denote a comment line with an asterisk in position 1.
- Input an option associated with a keyword when you specify the keyword. To obtain the total default value of the keyword, omit the entire keyword and option.

Override Defaults

You can override option default values by specifying options in the card-image data set defined by the CAIOPTS DD statement. You can specify this DD statement at execution time.

At view time, you can override options through the Report Options panel of the CAIPRINT Repository Viewer. For more information, see [Using the CAIPRINT Repository Viewer \(see page 263\)](#).



Note: Any options that were fixed in the installation defaults cannot be changed. The product is shipped with all defaults unfixed.

CA SymDump Batch JCL Requirements

By default, CA SymDump Batch does not require any additional JCL at execution time. All CA SymDump Batch modules must be made available at installation time (through LPA or LINKLIST), and CA SymDump Batch is initialized using CA Common Services for z/OS. For more information about initializing CA SymDump Batch, see [Installing \(https://docops.ca.com/display/CAITSD11/Installing\)](https://docops.ca.com/display/CAITSD11/Installing).

Because CA SymDump Batch remains dormant until a dump is requested as the result of an ABEND, you must include a SYSUDUMP DD statement in your JCL to receive abend reports. To suppress the system dump, specify DUMP OFF in your options file overrides, or DUMP=OFF in your installation defaults (CAOETABL).

- [Repository Requirements \(see page 200\)](#)
- [Report File Requirements \(see page 200\)](#)
- [Option File Requirements \(see page 200\)](#)
- [Symbolic File Requirements \(see page 200\)](#)

Repository Requirements

The repository used by CA SymDump Batch has a default name of CAIPRTLB, which can be customized at installation time. This file is dynamically allocated, if not included in the JCL, using the data set name specified at installation time. You can use the user exit CAOCUPRT to select a repository library based on criteria such as the job name or userid. You can also override the default repository library by specifying the PRTLIB execution option.

Report File Requirements

The report file used by CA SymDump Batch has a default name of CAIPRINT, which you can change during installation. This file is dynamically allocated if it is not included in the JCL. The SYSOUT class for dynamic allocation is also an installation option with a default value of SYSOUT=*.

Option File Requirements

The options file has a default name of CAIOPTS, which you can change during installation. This file can also be dynamically allocated if it is not included in the JCL at execution time. The high-level qualifier of the options file for dynamic allocation is equal to the userid, while the remainder of the data set name is selected at installation time. The default name is userid.CAIOPTS.

Symbolic File Requirements

The symbolic file execution-time override has a default name of CAISYM, which you can change during installation. This file is not required in the JCL, but can be used to augment the symbolic file selection list defined at installation time. If a symbolic file override is specified in the JCL or through the SYMDSN execution option, it is searched first for symbolic information. This lets each user or group of users maintain their own symbolic file.

Suppress Abend Reports

In addition to the exclusion criteria provided with the default member CAOETABL, you can suppress an abend report for any job step by adding a DD statement to your JCL.

To suppress abend reports, add a DD statement to your JCL.

The STOPDD option, defined in CAOETABL, defines the name of the DD. The default value is CAOESTOP. Following is an example of using this DD with the default value:

```
//CAOESTOP DD DUMMY
```

When the DD statement specified in your defaults table is present, CA SymDump Batch is bypassed and your standard dump facility is invoked.

Reports

Four reports are described in this section. The Execution Monitor produces two reports, the CAIOPTS File Processing Report and the Execution Monitor Summary. The Execution Monitor is the component of CA SymDump Batch that establishes the reporting environment. The CAIOPTS File Processing report displays the input options you specified to control abend or Snap reports. The Execution Monitor Summary displays the options in effect at the time the report was generated.

The following list describes the abend and Snap reports:

- The abend report is printed whenever an ABEND occurs during program execution. The report contains the following:
 - Diagnostic information based on the type of ABEND.
 - The COBOL, PL/I, or Assembler source statement on which the ABEND occurred, if symbolic information for the abending program was available. Otherwise, the machine instruction and its operands are displayed.
 - The names and values of the data items referenced by the abending statement are also displayed if symbolic information is available from a PROTSYM file, or from a CSL file if MXREF was specified when the CSL member was created.
 - Addresses and other specifics pertinent to the ABEND.
 - The module call sequence showing the hierarchy of programs that invoked the abending program.
 - The contents of the abending program's registers at ABEND.

- Formatted data displays of all program variable fields, if symbolic support is available. For COBOL, this includes Working Storage, Local-Storage, Linkage and File Sections. For PL/I, this includes all variables for all active procedures and a dump of both static and external program storage. For Assembler, this includes storage mapping of all addressable storage from the USINGs in effect at the abending statement and a dump of all unmapped addressable storage. If symbolic support is not available, all data areas are displayed in standard dump format.
- A standard dump display of the program-level control blocks. For COBOL, this includes the PGT, TGT, and DSA. For PL/I, this includes the DSA for each active procedure.
- An optional display of the memory map (TGT, PGT, DSA).
- The data control blocks of all open files not reported on in the FILE SECTION (optional).
- The Save Area Trace report showing the standard chain of register areas (optional).
- The Snap report provides you with a snapshot of your program at strategic points during execution. The Snap report contains the same information as the abend report with the exception of the data identifying the ABEND and the module call sequence. To produce a Snap report, you must do the following:
 - Insert a call to program CAODSNAP.
 - Specify the SNAP ON option at execution time.



Note: Each call to CAODSNAP generates a separate Snap report in your central VSAM repository. If your program contains multiple snap calls (or a snap call within a loop), you may want to write the output to the CAIPRINT DD alone, to avoid *flooding* the repository. For more information, see the PRTREPT option in [Reporting Options \(see page 190\)](#).

Abend Report

- [Abend Page \(see page 204\)](#)
- [The Offending Instruction \(see page 208\)](#)
- [Module Call Sequence \(see page 209\)](#)
- [COBOL Reports \(see page 210\)](#)
- [PL/I Reports \(see page 218\)](#)
- [Assembler Reports \(see page 225\)](#)
- [Open Files Report \(see page 231\)](#)
- [Save Area Trace Report \(see page 232\)](#)
- [IMS Report \(see page 233\)](#)
- [DB2 Report \(see page 242\)](#)
- [CA IDMS/DB Report \(see page 249\)](#)

The CA SymDump Batch abend report contains the following sections:

- Abend page, including abending source statement

- Module call sequence
- Program information for each active program or procedure
- Open Files report
- Save Area Trace report
- IMS report
- DB2 report
- CA IDMS/DB report

Program information is provided for each active program or procedure at the time of the ABEND. The content depends on the programming language, as follows:

- COBOL
 - Compile and link statistics
 - Data division displays
 - Registers
 - PGT, TGT, and DSA
- PL/I
 - Compile and link statistics
 - Last statement executed in each active procedure
 - Variables for each active procedure
 - Last registers for each active procedure
 - DSA for each active procedure
 - Static and external storage dump
 - Program storage dump
- Assembler
 - Link statistics
 - Program Status Word
 - Registers, including access registers
 - Addressable storage displays

■ Program storage dump

Abend Page

The abend report begins with the abend page, which displays specific information pertaining to the ABEND. A sample abend page follows:

```

*****
*           * S-0C7 *
*   A B E N D   *-----*
*           * CAOEDMO *
*****

*****
* CEE3207S The system detected a data exception (System Completion      *
*           Code=0C7). *
*****
****
*
*   DESCRIPTION:  S0C7 - DATA EXCEPTION: A NUMERIC FIELD CONTAINED NON-
NUMERIC          *
*
*               DATA.
*
*
*
*   POSSIBLE CAUSES:  1. NUMERIC DATA WAS NOT INITIALIZED, OR VARIABLE DATA NOT
*
*                     NUMERIC.
*
*                     2. ATTEMPTED TO PERFORM A DECIMAL ARITHMETIC INSTRUCTION ON
*
*                     EITHER AN UNSIGNED OR UNPACKED FIELD.
*
*                     3. A SUBSCRIPT OR INDEX CONTAINED AN INVALID VALUE:
*
*                         A. FAILURE TO INITIALIZE A SUBSCRIPT.
*
*                         B. INITIALIZED A SUBSCRIPT TO ZERO, BUT FAILED TO ADD 1
*
*                         BEFORE ITS FIRST USE.
*
*                         C. NOT REINITIALIZING A SUBSCRIPT AFTER A LOOP USING IT.
*
*                         D. SUBSCRIPTING BY INPUT DATA THAT WAS NOT CHECKED FIRST FOR
*
*                         A VALID RANGE OF VALUES.
*
*                         E. COMPUTING THE SUBSCRIPT WITHOUT CHECKING THE RESULTS FOR
*
*                         A VALID RANGE OF VALUES.
*
*
*   TO CORRECT:  1. REVIEW AND CORRECT THE ABOVE STATED POSSIBLE CAUSES.
*
*                2. REVIEW THE LAST KNOWN I
/O OR CALL INFORMATION. IT IDENTIFIES *
*                THE PROGRAM AT THE OFFSET WHERE THE LAST KNOWN I
/O OR CALL WAS *
*                MADE.
*
*                3. REVIEW THE Z
/O OR CALLING EVENT ADDRESS LINE. IF PRESENT, IT *
*                IDENTIFIES THE PROGRAM AT THE OFFSET WHERE THE BRANCH
*
*                ORIGINATED.
*
*****
****
*****

```

* ABENDING STATEMENT *

PROGRAM: CAOEDEMO OFFSET: 003D78 LINE: 001285

001285 MOVE MASK1 TO BINARY-1

LINE# /LOCATION	LEVEL/FIELD NAME DEFINITION	VALUE	
000329	03 MASK1	ALL '*'	X(2)
000178	03 BINARY-1	+1	S9
(4) BINARY			

ABENDING INSTRUCTION
 4F40 D4B8 CONVERT

OPERAND: REG 4
 VALUE: 00000002

OPERAND: TEMP STORAGE CELL DSA DISP 0004B8
 VALUE: 000000000000CC5 <-- NOT NUMERIC

INTERRUPT OCCURRED AT ADDRESS B6304686 AT OFFSET +003D7E IN PROGRAM CAOEDEMO

ENTRY POINT ADDRESS IS 36300908 AT OFFSET +000000 IN PROGRAM CAOEDEMO

LAST KNOWN I

/O OR CALL OCCURRED AT ADDRESS 36301CDE AT OFFSET +0013D6 IN PROGRAM CAOEDEMO

The abend page begins with the abend code and associated diagnostic information:

```

*****
*           * S-0C7 * [A]
*   A B E N D *-----*
*           * CAOEDEMO * [B]
*****
*****
* CEE3207S The system detected a data exception (System Completion Code=0C7). *
* [C]
*****
*****
* [D] DESCRIPTION: S0C7 - DATA EXCEPTION: A NUMERIC FIELD CONTAINED NON-
NUMERIC
*
*
*
* POSSIBLE CAUSES: 1. NUMERIC DATA WAS NOT INITIALIZED, OR VARIABLE DATA NOT
*
* [E] NUMERIC.
*
*
* 2. ATTEMPTED TO PERFORM A DECIMAL ARITHMETIC INSTRUCTION ON
*
* EITHER AN UNSIGNED OR UNPACKED FIELD.
*
*
* 3. A SUBSCRIPT OR INDEX CONTAINED AN INVALID VALUE:
*
* A. FAILURE TO INITIALIZE A SUBSCRIPT.
*
* B. INITIALIZED A SUBSCRIPT TO ZERO, BUT FAILED TO ADD 1
*
* BEFORE ITS FIRST USE.
*
* C. NOT REINITIALIZING A SUBSCRIPT AFTER A LOOP USING IT.
*
* D. SUBSCRIPTING BY INPUT DATA THAT WAS NOT CHECKED FIRST FO
R
*
* A VALID RANGE OF VALUES.
*

```

```

*
*           E. COMPUTING THE SUBSCRIPT WITHOUT CHECKING THE RESULTS FOR
*
*           A VALID RANGE OF VALUES.
*
*           4. A COMP-
3 FIELD HAD AN INVALID SIGN.
*
*
* [F]      TO CORRECT:  1. REVIEW AND CORRECT THE ABOVE STATED POSSIBLE CAUSES.
*
*           2. REVIEW THE LAST KNOWN I
/O OR CALL INFORMATION. IT IDENTIFIES
*           THE PROGRAM AT THE OFFSET WHERE THE LAST KNOWN I
/O OR CALL WAS
*           MADE.
*
*           3. REVIEW THE Z
/O OR CALL INFORMATION. IT IDENTIFIES
*           THE PROGRAM AT THE OFFSET WHERE THE BRANCH
*           ORIGINATED.
*
*****
*****

```

- **[A]**
Displays the system or user abend code.
- **[B]**
Displays the abending module name.
- **[C]**
Displays the LE message associated with the ABEND, if LE is active and a message is available.
- **[D]**
Displays a description of the abend code.
If the abend code indicates a DB2 ABEND (for example, S04E or S04F), the DB2 reason code is displayed with the appropriate description.
- **[E]**
Displays a list of possible causes for the ABEND.
- **[F]**
Displays suggested strategies for solving the problem.

If the ABEND occurs in a DB2 application, the DB2 return code, SQLCODE, is displayed with the associated messages.

The abending statement information follows the diagnostic box:

```

*****
* ABENDING STATEMENT *
*****

[A] PROGRAM: CAOEDMO  OFFSET: 003D78  LINE: 001285

001285      MOVE MASK1  TO  BINARY-1 [B]

LINE#   LEVEL/FIELD NAME      VALUE
/LOCATION      DEFINITION
-----
[C] 000329  03 MASK1          ALL '*'

```

```
(2) 000178 03 BINARY-1 +1 S9
(4) BINARY
```

- **[A]**
Displays the name of the program that contains the abending statement, the offset into the program for that statement, and line number of the statement.
If the program was compiled using PL/I, this line is followed by a line displaying the name of the procedure that contains the abending statement and the offset into the procedure for that statement.
- **[B]**
If symbolic information is available for the program, displays the abending COBOL, PL/I, or Assembler source statement, including the line number from the source listing.
The LISTLINE option lets users provide an additional context for the source statement at ABEND, snap, or transfer. The value of the option is used to determine the number of additional source listing lines to display before and after the abending, snap, or transfer statement. If the LISTLINE option is nonzero, the specified number of source listing lines leading up to the statement is displayed. This is followed by up to 10 lines of the abending, snap, or transfer statement (including comment lines), with the first line delineated by dashed lines. Then the specified number of source listing lines following the statement is displayed. If the LISTLINE option is zero, up to 10 lines of the abending, snap, or transfer statement are displayed (excluding comment lines for COBOL and Assembler programs).
- **[C]**
Displays the COBOL or PL/I variable names referenced by the abending source statement. This is available only if symbolic information is obtained from a PROTSYM file, or from a CSL if the XREF option was used when the CSL member was created.

The abending instruction is then displayed, followed by its operands. For COBOL and PL/I programs, this information is suppressed when the ASMINST option is set to OFF:

```
ABENDING INSTRUCTION
[A] 4F40 D4B8          CONVERT

OPERAND: REG 4
[B] VALUE: 00000002

OPERAND: TEMP STORAGE CELL      DSA      DISP 0004B8
VALUE: 00000000000000CC5      [C] <-- NOT NUMERIC
```

- **[A]**
Displays the machine instruction that caused the ABEND.
- **[B]**
Displays the instruction operands for the abending instruction. If possible, the operands are related back to an area in the abending program.
- **[C]**
Displays the specific cause for the ABEND, if known.

The abend page ends with address information that may be helpful in debugging the ABEND.

```
INTERRUPT OCCURRED AT ADDRESS B6304686 AT OFFSET +003D7E IN PROGRAM CAOEDemo [A]

ENTRY POINT ADDRESS IS 36300908 AT OFFSET +000000 IN PROGRAM CAOEDemo [B]
LAST KNOWN I
```

/O OR CALL OCCURRED AT ADDRESS 36301CDE AT OFFSET +0013D6 IN PROGRAM CAOEDMO [C]
THE Z/OS BREAKING EVENT INFORMATION IS NOT AVAILABLE. [D]

- **[A]**
Displays the absolute address and program offset where the interrupt occurred. If the abending program is PL/I, the procedure offset is also displayed.
- **[B]**
Displays the absolute address and program offset of the entry point for the abending program.
- **[C]**
Displays the absolute address and program offset from which the last known I/O or CALL was made. If the abending program is PL/I, the procedure offset is also displayed. Alternatively, if the last I/O or CALL information is not available, that is stated here.
- **[D]**
Displays the absolute address and program offset of the last z/OS breaking event (from the breaking-event-address register). If the abending program is PL/I, the procedure offset is also displayed. Alternatively, if the breaking event information is not available, that is stated here.

The Offending Instruction

In the case of program check interruptions (OCx), CA SymDump Batch reports on the particular instruction that caused the ABEND with as much supporting information as possible. CA SymDump Batch can provide additional information for the following:

- Protection
- Addressing
- Data
- Fixed-point overflow
- Fixed-point divide
- Decimal overflow
- Decimal divide
- Exponent overflow
- Significance
- Floating-point divide

CA SymDump Batch cannot always provide further analysis of the instruction, especially with the following exceptions:

- Operation
- Privileged operation
- Execution

- Specification

Module Call Sequence

The next block of information is important if the application has multiple program modules. It lists the calling sequence of programs that resulted in the execution of the program that produced the ABEND.

```
*****
* MODULE CALL SEQUENCE *
*****
```

	[A] [F] LOAD	[G] PROGRAM	[B] [H] VER/ ADDRESS	PROGRAM LENGTH	[C] PROGRAM ADDRESS	[D] ENTRY DATE	[E] COMPILED TIME	[E] COMPILED LANGUAGE
E	MODULE /PROCEDURE GE REL							
	-----	-----	-----	-----	-----	-----	-----	-----
		SYSTEM			00000000	000000		36300908
	ASSEMBLER CALLED CEE	CEE		00000000	000000	00000000		
		ASSEMBLER						
	CALLED CAOEDMO	CAOEDMO		36300908	004AA8	36300908		14
	MAR,2008 16.38.30	COB Z/OS 3.4.1						
	(ABENDED)							

The module call sequence portion displays the following information:

- **[A]**
Displays the load module name in which the called program resides.
- **[B]**
Displays the name of the called program. For COBOL and Assembler, this is the CSECT name. For PL/I this is the active procedure name.
- **[C]**
Displays the storage address of the start of the called program.
- **[D]**
Displays the length of the called program.
- **[E]**
Displays the storage address of the entry point of the called program.
- **[F]**
Displays the date and time of compilation.
- **[G]**
Displays the source language in which the called program is coded.
- **[H]**
Displays the version of the COBOL or PL/I compiler used for compilation.



Note: The register Save Area Trace portion is at the end of the abend report. If you want to know the actual contents of the registers when control was passed from program to program, [see the Save Area Trace Report](#).

COBOL Reports

This section describes the information provided for COBOL programs.

Program Statistics

For each program in the report, this report contains statistics about the compile, link, and symbolic postprocessing used when the program was built.

```
*****
* PROGRAM CAOEDMO *
*****
```

[A]

```
COMPILED ON 14 MAR,2008 AT 16.38.30 WITH COBOL FOR Z/OS 3.4.1
COMPILE OPTIONS:ADV, APOST, NOAWO, DATA(24), NODBCS, NODECK, NODUMP, DYNAM,
NOFASTSRT, NOLIB, LIST, MAP, NONAME, NONUMBER, OBJECT,
NOOFFSET, NOOPTIMIZE, NUMPROC(PFD), RENT, NOSEQUENCE, SOURCE,
NOSSRANGE, NOTERM, NOTEST, TRUNC(STD), NOVVBREF, NOWORD, XREF,
ZWB
```

[B]

```
SYMBOLIC INFORMATION RETRIEVED FROM USER01.PROTSYM
MEMBER: CAOEDMO DATE: 14 MAR,2008 TIME: 16.38.30 TYPE: PROTSYM
```

[C]

```
LINKED ON 14 MAR 2008 AT 16:38:33
LOAD LIBRARY: USER01.LOAD
MODULE LENGTH: 000056F8 (DECIMAL 22,264)
LINK OPTIONS: AC(0), AMODE(31), NOOVLY, NORENT, NOREUS, RMODE(24)
```

The COBOL report displays the following information for each program:

- **[A]**
Displays compile statistics, including the date and time of compilation, the compiler and release information, and the options used.
- **[B]**
Displays the name of the symbolic file and the date and time of the symbolic listing used when symbolic information is used to format the report.
- **[C]**
Displays link-edit information including the date and time when the load module was created, the name of the load library from which the program was loaded, the length of the module, and the linkage editor options used.

Data Division Displays

The Data Division portion of the abend report contains the following sections:

- File Section display
- Working-Storage Section display
- Local-Storage Section display

- Linkage Section display

File Section Display

The File Section portion of the abend report lists information for each file defined to the COBOL program, such as allocation information, access method, and whether the file is open or closed. If the file is open, information from the data management control blocks, the current file status, and the contents of the current logical record are displayed. If the file was accessed sequentially, the contents of the previous record may also be displayed. Optional displays include catalog information for VSAM files and dumps of data management control blocks.

The following example shows the beginning of the File Section portion, including the first four lines of information for an open QSAM file:

```
*****
* FILE SECTION *
*****

[A] DDNAME=PRINT      DSORG=PS   QSAM   BUFNO=5      UNIT=3390,UCB   VOL=SER=WRKD28
    DSN=USER02.
PRINT                                RECFM=FB
[B] STATUS(OPEN)      USAGE(OUTPUT)      LRECL=120      BLKSIZE=120
[C] CURRENT FILE STATUS = 00  SUCCESSFUL
```

The File Section portion displays the following information:

- **[A]**
Displays the data management control blocks. The data management blocks are interpreted and displayed in KEYWORD=value format. This information includes the ddname, data set organization, access method, number of buffers allocated, and data set name. For non-VSAM files, the unit type, volser and record format are also displayed.
- **[B]**
Indicates the file's current status (OPEN, CLOSED), usage (INPUT, OUTPUT, etc.), and the record length. For non-VSAM files, the block size is also displayed, while the feedback code, function code, and return code are displayed for VSAM files.
- **[C]**
Displays the hexadecimal value of the current file status along with a brief description of what it means.

For open VSAM files, the VSAM catalog information can be displayed using the VSAMCAT option. This option can be specified as an execution-time override, in the CAIOPTS DD. When the option value is set to ON, the VSAM catalog information is displayed following the Current File Status field, as shown in the following example:

```
DDNAME=INDEX1 DSORG=KSDS VSAM BUFNO=253
DSN=USER02.INDEX1
STATUS(OPEN)  USAGE(OUTPUT)      LRECL=240      FDBK=0,FTNCD=0,RC=0
CURRENT FILE STATUS = 00  SUCCESSFUL

[A] CLUSTER NAME      USER02.INDEX1
    DATA COMPONENT   USER02.INDEX1.DATA
    INDEX COMPONENT   USER02.INDEX1.INDX
[B] ACTIVITY DATES:
    CREATION DATE     2007.085      EXPIRATION DATE NEVER SCRATCH
[C] DATASET DEFINITION:
    KEY POSITION        128           KEY LENGTH      29
    AVERAGE RECORD SIZE 240         MAX RECORD SIZE 240
    SHARE OPTIONS      2,3          BUFFER SPACE 4608
```

```

[D] ALLOCATION PARAMETERS:
WRITE CHECK      NO  ERASE ON DELETE NO
REUSE OPTION      NO  SPANNED RECORDS NO
REPLICATED INDEX NO
[E] DATA COMPONENT INFORMATION:  F  INDEX COMPONENT INFORMATION:
VOLUME           OSI006           OSI006
ALLOCATION UNIT TRACKS           TRACKS
PRIMARY ALLOCATION 4             1
SECONDARY ALLOCATION 1           1
EXTENTS USED      1             1
HIGH ALLOCATED RBA 147456       23552
HIGH USED RBA     0             0
CI SIZE           2048          512
[G] STATISTICS:
RECORDS RETRIEVED 0  RECORDS DELETED 0
RECORDS INSERTED  0  RECORDS UPDATED 0
TOTAL RECORDS     250

```

The following information is displayed for open VSAM files:

- **[A]**
Displays the data set names of the cluster and data components. For KSDS files, the data set name of the index component is also displayed.
- **[B]**
Displays the date on which the file was created and the date on which it may be deleted. The dates are displayed in Julian format (YYYY.DDD). If the expiration date is displayed as *Never Scratch*, the PURGE parameter is always required to delete the file.
- **[C]**
Displays the key position and length, the average and maximum lengths of the data records, the share options, and the minimum buffer space allocated when the file is accessed.
The key position and length are only displayed for KSDS files and variable-length RRDS files. The key position value indicates the position of the key relative to the start of the record. For alternate index files, the key position and length refer to the alternate key within the base cluster. Otherwise, they refer to the prime key within the base cluster.
- **[D]**
Indicates whether or not the file was allocated using the WRITECHECK, ERASE, REUSE, SPANNED, and REPLICATE parameters.
- **[E]**
Displays information about the volume on which the file is allocated, the allocation unit (tracks or cylinders), the primary and secondary allocation amounts, the number of extents used, the highest allocated and highest used relative byte address, and the size of the control interval.
- **[F]**
Displays information for the same fields described under E. This information is displayed only for KSDS files.
- **[G]**
Displays the number of times data records are retrieved, deleted, inserted, or updated since the initial load of the file, as well as the total number of data records in the file.

The contents of the current logical record are displayed following the Current File Status field or, if the file is VSAM and the VSAMCAT option is ON, following the VSAM catalog information. The contents of the previous record may also be displayed for sequentially accessed files. For keyed VSAM files, the VSAM key value is displayed prior to the contents of the record. Symbolic information may be merged with the current record data (as shown in the following screen), if it is available.

CURRENT LOGICAL KEY

[A]

LINE#	LEVEL/FIELD NAME	VALUE	DEFINITION
000158	20 IX-FS1-KEY	* ABCDLKJXYZ000000250ZIF, .\$.+CD*	X

CONTENTS OF CURRENT LOGICAL RECORD

[B]

LINE#	LEVEL/FIELD NAME	VALUE	DEFINITION
000094	01 IX-FS1R1-F-G-240	BLF=0002+000000 (0000F078)	
000095	03 IX-FS1-WRK-120	*FILE=IX-FS1,RECORD=IX-F-G/0,RECN*	X
		(+000032) *0=000250,UPDT=00,OD0=0000,PGM=IX*	
		(+000064) *101,LRECL=000240,BLKSIIZRC=0001,L*	
		(+000096) *FIL=000500,ORG=IX,LBLR=S*	
000096	03 IX-FS1-GRP-120	BLF=0002+000078 (0000F0F0)	
000097	05 FILLER	*	
	RECKEY=*	X(8)	
000098	05 IX-FS1-KEY	*ABCDLKJXYZ000000250ZIF, .\$.+CD*	X
	(29)		
000099	05 FILLER	*	
	ALTKEY1=	X(83)	
		(+000032) *	
	ALTKEY2=	*	
		(+000064) *	

PREVIOUS KEY

[C]

OFFSET	LENGTH 0000001D (DECIMAL 29)
+000000	C1C2C3C4 D3D2D1E7 E8E9F0F0 F0F0F0F0 F2F4F9E9 C9C66B4B 5B6
04EC3 C4	*ABCDLKJXYZ000000249ZIF, .\$.+CD*

CONTENTS OF PREVIOUS RECORD

OFFSET	LENGTH 000000F0 (DECIMAL 240)
+000000	C6C9D3C5 7EC9E760 C6E2F16B D9C5C3D6 D9C47EC9 E760C660 C761
F06B D9C5C3D5	*FILE=IX-FS1,RECORD=IX-F-G/0,RECN*
+000020	D67EF0F0 F0F2F4F9 6BE4D7C4 E37EF0F0 6BD6C4D6 7EF0F0F0 F06B
D7C7 D47EC9E7	*0=000249,UPDT=00,OD0=0000,PGM=IX*
+000040	F1F0F16B D3D9C5C3 D37EF0F0 F0F2F4F0 6BC2D3D2 E2C9E9D9 C37E
F0F0 F0F16BD3	*101,LRECL=000240,BLKSIIZRC=0001,L*
+000060	C6C9D37E F0F0F0F5 F0F06BD6 D9C77EC9 E76BD3C2 D3D97EE2 6BD9
C5C3 D2C5E87E	*FIL=000500,ORG=IX,LBLR=S,RECKEY=*
+000080	C1C2C3C4 D3D2D1E7 E8E9F0F0 F0F0F0F0 F2F4F9E9 C9C66B4B 5B60
4EC3 C46BC1D3	*ABCDLKJXYZ000000249ZIF, .\$.+CD,AL*
+0000A0	E3D2C5E8 F17E4040 40404040 40404040 40404040 40404040 4040
4040 40404040	*TKEY1=
+0000C0	4040406B C1D3E3D2 C5E8F27E 40404040 40404040 40404040 4040
4040 40404040	* ,ALTKEY2=
+0000E0	40404040 40404040 40404040 40404040

The following information is displayed for the current logical record and the previous record:

- **[A]**
Displays the current logical key data only for VSAM files with defined keys (i.e. KSDS and variable-length RRDS files). If the MRGFILES option is on, the data may be displayed in merged format. Otherwise, the data is displayed in dump format.
- **[B]**
Displays the contents of the current logical record using the following options:
 - If the FIRST128 option is specified, only the first 128 bytes of the record are displayed.
 - If the MRGFILES option is specified, the data is displayed in merged format. Otherwise, the data is displayed in dump format.
- **[C]**
Displays previous record data for sequentially accessed files (VSAM, QSAM, and BSAM), if it is available in an I/O buffer and:
 - The file is not a BSAM file using NOTE/POINT
 - The file is not a VSAM file using skip-sequential access
 - The file is not a JES managed data set, such as in-stream data or data written to a SYSOUT class
 - The file does not contain unformatted, spanned, compressed, or extended records

For keyed VSAM files, the previous record key is displayed prior to the contents of the previous record.

Use the Data Management Control Block options (DEB, DCB, UCB, and IOB) or VSAM Control Block options (ACB, AMBL, AMB, PLH, and RPL) to dump control blocks after the record data display. To dump all of the control blocks, specify the BLOCKS option. These options may be specified as execution-time overrides, in the CAIOPTS DD.

The following example shows the Data Management Control Blocks for a QSAM file displayed in hexadecimal format.

```

DATA MANAGEMENT CONTROL BLOCKS
-----
DEB      007C2D84      LENGTH 00000088  (DECIMAL 136)
          OFFSET
          +000000      037C4E88  10000000  E8000000  0F001100      01000000  FF000000  8F00
D038 047C2D60
          +000020      187CED48  00000002  00030002  00030001      00010001  00000000  0000
0000 00000079
          +000040      F3C2C1D9  C1C90000  00000000  00000000      00000000  00000000  0000
0000 00000000
          +000060      00000000  00540002  007C7808  00001BE2      00000000  02002000  0000
0000 00000000
          +000080      007AD998  C4E2C3C2

DCB      0000D038      LENGTH 00000060  (DECIMAL 96)
          OFFSET
          +000000      22A8CA50  00000000  00020003  13F53026      002FAB98  05018FE8  0000
4000 00006C08
          +000020      C6000001  840138B4  00A40048  007C2D84      92D8C2C0  00D8BE70  0A01

```

```

272A 02090079
      +000040 30013030 00006DB8 22ABAFF9 22ABAFF9 00000079 00000000 0000
0000 00E6E470

UCB    007CED48      LENGTH 00000040 (DECIMAL 64)
      OFFSET
      +000000 01A8FF8C 2D050000 00000000 08E4C3C2 3030200F 00229D21 5DC0
0100 E6D9D2C4
      +000020 F2F51001 00A00014 02229B48 02233188 05800101 00000000 4068
8072 20C69F40

IOB    00006C08      LENGTH 00000050 (DECIMAL 80)
      OFFSET
      +000000 7F000000 007C1BD0 00006C78 80006DB8 00006C78 04040004 0000
0000 00000000
      +000020 00000000 00000000 00000000 00000000 00000000 00000000 0000
0000 00000000
      +000040 00000000 00000000 00000000 00000000

```

Working-Storage Section, Local-Storage Section, Linkage Section Displays

The abend report displays the Working-Storage Section, Local-Storage Section, and the Linkage Section. An example of a Linkage Section Display follows:

```

*****
* LINKAGE SECTION *
*****

```

LINE# /LOCATION	LEVEL/FIELD NAME	DEFINITION	VALUE
000399	01 PARM		BLL=0001+000000 (36310E68)
000400	03 PARM-		
	LENGTH	+1	S9
	(4) BINARY		
000401	03 PARM-		
	DATA	? X'E8'	X(12)

LINE# /LOCATION	LEVEL/FIELD NAME	DEFINITION	VALUE
001353	01 INPUT-PARM		BLL=0002+000000 (365864B0)
001354	03 INPUT-PARM-		
	LENGTH	+1	S9(4) BINARY
001355	03 INPUT-PARM-DATA	? X'B38C000000000000000000000000'	X
	(12)		

If the MRGWORK (for Working-Storage), MRGLOCAL (for Local-Storage), and MRGLINK (for Linkage Section) options are ON, the data is displayed in merged format. Otherwise, it is displayed in dump format.

For more information about the merged and dumped data formats, see [Merged Versus Dumped Data Display](#) (see page 259).

Register Contents

The registers at abend portion of the abend report displays the general registers, floating-point registers and vector registers for the abending program:

```

*****
* REGISTERS AT ABEND *
*****

```

```

[A]
GENERAL REGISTERS:

```

```

R0      36586520      DISP +A99B50 IN BLL 0002
R1      365864A8      DISP +000478 IN DSA
R2      0001D0B8      DISP +000000 IN BLW 0000
R3      000360B8      DISP +000000 IN BLW 0019
R4      00000002
R5      B630409E      DISP +003796 IN PROGRAM CAOEDMO
R6      36301C76      DISP +00136E IN PROGRAM CAOEDMO
R7      00000000
R8      000340B8      DISP +000000 IN BLW 0017
R9      0001B038      DISP +000000 IN TGT
R10     36300A70      DISP +00006C IN PGT
R11     3630456E      DISP +003C66 IN PROGRAM CAOEDMO
R12     36300A04      DISP +000000 IN PGT
R13     36586030      DISP +000000 IN DSA
R14     B6301CE0      DISP +0013D8 IN PROGRAM CAOEDMO
R15     00000000

```

[B]

FLOATING-POINT REGISTERS:

FPC	00000000						
FR0	493202EA	E0000000	1.342483 E+10	FR8	00000000	00000000	0.0
FR1	00000000	00000000	0.0	FR9	00000000	00000000	0.0
FR2	4E000000	03923FD1	5.991624 E+07	FR10	00000000	00000000	0.0
FR3	00000000	00000000	0.0	FR11	00000000	00000000	0.0
FR4	4E000000	00025EF4	155380	FR12	00000000	00000000	0.0
FR5	00000000	00000000	0.0	FR13	00000000	00000000	0.0
FR6	00000000	00000000	0.0	FR14	00000000	00000000	0.0
FR7	00000000	00000000	0.0	FR15	00000000	00000000	0.0

[C]

VECTOR REGISTERS:

VR0	493202EA	E0000000	00000000	00000000	VR16	00000000	00000000	00000000	00000000
VR1	00000000	00000000	00000000	00000000	VR17	00000000	00000000	00000000	00000000
VR2	4E000000	03923FD1	00000000	00000000	VR18	00000000	00000000	00000000	00000000
VR3	00000000	00000000	00000000	00000000	VR19	00000000	00000000	00000000	00000000
VR4	4E000000	00025EF4	00000000	00000000	VR20	00000000	00000000	00000000	00000000
VR5	00000000	00000000	00000000	00000000	VR21	00000000	00000000	00000000	00000000
VR6	00000000	00000000	00000000	00000000	VR22	00000000	00000000	00000000	00000000
VR7	00000000	00000000	00000000	00000000	VR23	00000000	00000000	00000000	00000000
VR8	00000000	00000000	00000000	00000000	VR24	00000000	00000000	00000000	00000000
VR9	00000000	00000000	00000000	00000000	VR25	00000000	00000000	00000000	00000000
VR10	00000000	00000000	00000000	00000000	VR26	00000000	00000000	00000000	00000000
VR11	00000000	00000000	00000000	00000000	VR27	00000000	00000000	00000000	00000000
VR12	00000000	00000000	00000000	00000000	VR28	00000000	00000000	00000000	00000000
VR13	00000000	00000000	00000000	00000000	VR29	00000000	00000000	00000000	00000000
VR14	00000000	00000000	00000000	00000000	VR30	00000000	00000000	00000000	00000000
VR15	00000000	00000000	00000000	00000000	VR31	00000000	00000000	00000000	00000000

The registers at abend portion displays the following information:

▪ **[A]**

Displays the base address and displacement if the contents of a general register can be related directly to a base address and displacement.

For COBOL II, COBOL/370, COBOL for MVS and VM, COBOL for OS/390 and VM, or Enterprise COBOL for z/OS and OS/390, the base addresses may appear as the following:

BLF nnnn	Base Locator for FILE Section
BLW nnnn	Base Locator for WORKING-STORAGE Section
BLL nnnn	Base Locator for LINKAGE Section
BLK nnnn	Base Locator for LOCAL-STORAGE Section

BLX nnnn	Base Locator for externally located data
progrname	Signature code that starts at relative location 0 of the program
TGT	Task Global Table
PGT	Program Global Table
DSA	Dynamic Save Area

For OS/VS COBOL the base address may appear as:

BL
BLL
INIT1
TGT

A blank line following the general register display indicates that the origin of the register contents cannot be determined.

- **[B]**
Displays the floating-point control register and floating-point registers 0 through 15 if the basic floating-point extensions of z/OS are installed on your system. Otherwise, the floating-point registers 0, 2, 4, and 6 are displayed (as shown).
- **[C]**
Displays the vector registers 0 through 31 if the vector facility for z/Architecture is installed on your system.

Memory Map Display

The following section displays the memory map, which consists of the TGT, PGT, and for some releases of COBOL, the DSA. For COBOL II or COBOL/370, the PGT display includes the Constant Global Table (CGT). For COBOL for MVS and VM, COBOL for OS/390 and VM, and Enterprise COBOL for z/OS and OS/390, the display includes the DSA. To interpret the Memory Map Display, see the listing of the memory maps printed after the compiler source listing.

```
*****
* MEMORY MAP *
*****
DSA      36586030      LENGTH 000004F0 (DECIMAL 1,264)
ADDRESS  OFFSET
36586030 000000 00104001 36313360 00000000 B6301CE0 00000000 36586520 365864A8 00
0157FC *...-.....\.....y...*
36586050 000020 0001B38C 00000000 B630409E 36301C76 00000000 000340B8 0001B038 36
300A70 *.....*
36586070 000040 3630456E 36300A04 00000000 36586520 00000000 00000000 36586030 00
01B038 *...>.....-.....*
36586090 000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
365860B0 000080 00000000 21000000 00000003 00000000 B6301CE0 00000000 36586520 36
5864A8 *.....\.....y*
365860D0 0000A0 36304214 363041F8 00000002 B630409E 36301C76 00000000 000340B8 00
01B038 *.....8.....*
365860F0 0000C0 36300A70 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
36586110 0000E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
          LINES 00000100-00000140 SAME AS ABOVE
.
.
.
```

```

TGT      0001B038      LENGTH 000003E8 (DECIMAL 1,000)
ADDRESS  OFFSET
0001B038 000000      00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
0001B058 000020      00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
0001B078 000040      00000000 00000000 F3E3C7E3 00000000 06000000 64020260 36582100 00
0157FC *.....3TGT.....*
0001B098 000060      0001B420 00000001 00084927 00000000 00000000 0001B930 00000000 00
000000 *.....*
0001B0B8 000080      363129C0 000003E8 00000000 00000000 00000002 00000005 E2E8E2D6 E4
E34040 *...{...Y.....SYSOUT *
0001B0D8 0000A0      C9C7E9E2 D9E3C3C4 00000000 00000000 00000000 00000000 00000000 00
000000 *IGZSRTCD.....*
0001B0F8 0000C0      00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 *.....*
0001B118 0000E0      00000000 00000000 36300A04 00000000 0001B3BC 365823F0 36301107 00
000000 *.....0.....*
.
.
.
PGT      36300A04      LENGTH 00000A60 (DECIMAL 2,656)
ADDRESS  OFFSET
36300A04 000000      05F5E100 3B9ACA00 000186A0 00002710 00000001 00000000 40404040 40
404040 *..5.....f.....*
36300A24 000020      40404040 40404040 40404040 40404040 40404040 40400000 00000000 00
000000 *.....*
36300A44 000040      00000000 000C0000 0000000C 0000000F F0F0F0F0 F0C00000 36300A70 36
301504 *.....00000{.....*
36300A64 000060      36302556 36303564 3630456E 36301552 3630183C 36301DFC 363020EA 36
302196 *.....>.....O*
36300A84 000080      36302376 36302556 363025F2 3630276C 363028F0 36302A46 36302AC4 36
302C0A *.....2...%...0.....D.....*
36300AA4 0000A0      36302C28 36302E0E 36302EC0 36302F14 36303068 363030AE 363036E8 36
3036A4 *.....{.....Y...U*
36300AC4 0000C0      36303B68 36303C12 363040B2 36303CB2 36303CF6 36303D4A 36303D9E 36
303DF2 *.....6...0.....2*
36300AE4 0000E0      36303E46 36303E9A 363037E6 3630385C 36303914 363039DC 36303A22 36
303A76 *.....W...*.....*

```

PL/I Reports

This section describes the information provided for PL/I programs.

Program Statistics

For each program in the report, this report contains statistics about the compile, link, and symbolic postprocessing used when the program was built.

```

*****
* PROGRAM PLITEST1 *
*****
[A]
COMPILED ON 20 APR,2004 AT 11.49.52 WITH PL/I FOR Z/OS 3.2.0
[B]
SYMBOLIC INFORMATION RETRIEVED FROM CDE.DEVL.SB21.PROTSYM
MEMBER: PLITEST DATE: 20 APR,2004 TIME: 11.49.52 TYPE: PROTSYM
[C]
LINKED ON 20 APR 2004 AT 11:49:58
LOAD LIBRARY: USER001.LOAD
MODULE LENGTH: 000047C0 (DECIMAL 18,368)

```

The program statistics include the following information:

- **[A]**
Displays compile statistics, including the date and time of compilation, the compiler and release information, and the options used.

- [B]
When symbolic information is used to format the report, this section displays the name of the symbolic file and the date and time of the symbolic listing used.
- [C]
Displays link-edit information including the date and time when the load module was created, the name of the load library from which the program was loaded, the length of the module, and the linkage editor options used.

Active Procedures

When one or more PL/I programs are active at the time of an ABEND, CA SymDump Batch reports on each active PL/I procedure. The information provided for each procedure includes the following:

- Variables defined to the procedure
- Registers when last in control
- Dynamic Save Area (DSA)

Variables

The following sample report shows the variable display for an active PL/I procedure:

```
*****
* VARIABLES *
*****

DCL#    LVL FIELD NAME          VALUE
/LOCATION          ATTRIBUTES
-----
000365    PIC_INIT              ?    X'00000000000000000000000000000000'    AUTOMA
TIC CHARACTER(31)
                                (+000016) X'00000000000000000000000000000000'
000368    01 TTP                ADDRESS=12268BD1    STRUCT
URE AUTOMATIC
000368    02 TTPT                *    PPPP TABLE TP    PPPP
*    CHARACTER(32)
000368    02 TP_PICTURE_TABLE    STRUCTURE DIM(18)
(1)    ADDRESS=12268BF1
000368    03 TPICTURE            **
(1)
(31) VARYING
000368    03 TDEC                ?    X'0000'    FIXED DEC
(1)
IMAL(3)
000368    03 TBIN                +0    FIXED BIN
(1)
ARY(15)
<UNMERGED DATA FOR TP_PICTURE_TABLE>
                                X'00000000000000000000000000000000'
LENGTH 646
                                LINES 000016-
000624 SAME AS ABOVE
                                (+000640) X'000000000000'
000375    TTP2                  +0    AUTOMATIC
(1)
DIM(18) FIXED BINARY(15)
                                <UNMERGED DATA FOR TTP2>
                                X'00000000000000000000000000000000'
LENGTH 34
                                (+000016) X'00000000000000000000000000000000'
```

```
(+000032)      X'0000'
```

Variable storage displays information in merged format or in dump format, depending on your option settings. The following options control merging for PL/I variables:

- MRGAUTO -- Merges symbolic names onto PL/I automatic storage.
- MRGBASED -- Merges symbolic names onto PL/I based variable storage.
- MRGCNTLD -- Merges symbolic names onto PL/I controlled storage.
- MRGPARDS -- Merges symbolic names onto PL/I parameter storage.
- MRGSTAT -- Merges symbolic names onto PL/I static storage.

For more information about the merged and dumped data formats, see [Merged Versus Dumped Data Displays \(see page 259\)](#).

Registers

For each procedure, the values of the general-purpose registers at the time when the procedure was last in control are displayed.

For abending procedures, these are the current register values, and the floating-point and vector registers are also included. For all other procedures, these are the registers when the procedure last transferred control to another procedure or program.

The following sample report shows the PL/I procedure registers:

```
*****
*  REGISTERS AT ABEND *
*****
```

GENERAL REGISTERS:

```
R0  12268EA0
R1  122683C8
R2  00000024
R3  12102D9A   DISP +00003A IN PROCEDURE INIT_TP
R4  00000000
R5  121035E8   DISP +000000 IN STATIC
R6  12102FD8   DISP +000278 IN PROCEDURE INIT_TP
R7  00000001
R8  000002CC
R9  1228A868
R10 12103604   DISP +00001C IN STATIC
R11 122683C8
R12 12113650
R13 12268A78   DISP +000000 IN DSA
R14 12268EA0
R15 92102EF0   DISP +000190 IN PROCEDURE INIT_TP
```

FLOATING-POINT REGISTERS:

```
FPC F0000000
FR0 41100000 00000000 1.0      FR8 00000000 00000000 0.0
FR1 00000000 00000000 0.0      FR9 00000000 00000000 0.0
FR2 34000000 00000000 0.0      FR10 00000000 00000000 0.0
FR3 00000000 00000000 0.0      FR11 00000000 00000000 0.0
FR4 5B4EE2D6 D415B85A 1.0 E+32 FR12 00000000 00000000 0.0
FR5 00000000 00000000 0.0      FR13 00000000 00000000 0.0
FR6 4DCEF810 00000000 3.641037 E+15 FR14 00000000 00000000 0.0
```

FR7 00000000 00000000 0.0 FR15 00000000 00000000 0.0

VECTOR REGISTERS:

VR0	41100000	00000000	00000000	00000000	VR16	00000000	00000000	00000000	00000000
VR1	00000000	00000000	00000000	00000000	VR17	00000000	00000000	00000000	00000000
VR2	34000000	00000000	00000000	00000000	VR18	00000000	00000000	00000000	00000000
VR3	00000000	00000000	00000000	00000000	VR19	00000000	00000000	00000000	00000000
VR4	5B4EE2D6	D415B85A	00000000	00000000	VR20	00000000	00000000	00000000	00000000
VR5	00000000	00000000	00000000	00000000	VR21	00000000	00000000	00000000	00000000
VR6	4DCEF810	00000000	00000000	00000000	VR22	00000000	00000000	00000000	00000000
VR7	00000000	00000000	00000000	00000000	VR23	00000000	00000000	00000000	00000000
VR8	00000000	00000000	00000000	00000000	VR24	00000000	00000000	00000000	00000000
VR9	00000000	00000000	00000000	00000000	VR25	00000000	00000000	00000000	00000000
VR10	00000000	00000000	00000000	00000000	VR26	00000000	00000000	00000000	00000000
VR11	00000000	00000000	00000000	00000000	VR27	00000000	00000000	00000000	00000000
VR12	00000000	00000000	00000000	00000000	VR28	00000000	00000000	00000000	00000000
VR13	00000000	00000000	00000000	00000000	VR29	00000000	00000000	00000000	00000000
VR14	00000000	00000000	00000000	00000000	VR30	00000000	00000000	00000000	00000000
VR15	00000000	00000000	00000000	00000000	VR31	00000000	00000000	00000000	00000000

Dynamic Save Area

For each procedure, the Dynamic Save Area (DSA) is also displayed:

```
*****
* DYNAMIC SAVE AREA *
*****
```

```
DSA      12268A78      LENGTH 00000158 (DECIMAL 344)
ADDRESS  OFFSET
12268A78 000000      10000000 122683C8 00000000 92102E3C      9219A6C8 1210326C 12268B10 000
00000      *. . . . .CH. . . . .k. . . . .k.wH. . . . .% . . . . .*
12268A98 000020      12102D9A 12268B48 121035E8 12102FD8      00000001 12268B4C 1228A868 121
03604      *. . . . .Y. . . . .Q. . . . .<.y. . . . .*
12268AB8 000040      122683C8 00000000 00000000 12268EC8      00000000 00000000 00000000 000
00000      *. . . . .CH. . . . .H. . . . .<. . . . .*
12268AD8 000060      00000000 00000000 00000000 00000000      00000000 00000000 00000000 000
00000      *. . . . .<. . . . .<. . . . .<. . . . .*
12268AF8 000080      00000000 00000000 00000000 00000000      00000000 00000000 12268B4C 122
68B48      *. . . . .<. . . . .<. . . . .<. . . . .*
12268B18 0000A0      12268B60 1210326C 00000000 00000000      00000000 00000000 00000000 000
00000      *. . . . .% . . . . .% . . . . .% . . . . .*
12268B38 0000C0      00000000 00000000 12268EA0 12268BD1      00000001 000002CB 00000002 000
00002      *. . . . .J. . . . .J. . . . .J. . . . .*
12268B58 0000E0      00000012 00000001 00000000 00200002      00000020 00000026 00000026 000
00012      *. . . . .<. . . . .<. . . . .<. . . . .*
12268B78 000100      00000001 001F8002 00000041 00000026      00000026 00000012 00000001 000
00043      *. . . . .<. . . . .<. . . . .<. . . . .*
12268B98 000120      00000026 00000026 00000012 00000001      00000000 00000000 122683C8 122
68460      *. . . . .<. . . . .<. . . . .<. . . . .*
12268BB8 000140      00000000 00000000 00000000 00000000      0000001C 00000000
*. . . . .<. . . . .<. . . . .<. . . . .*
```

Storage

After all of the active procedures are reported, the program storage is displayed. This includes both static and external program storage areas.

Static Storage

Static storage is displayed in dump format as shown in the following screen:

```
*****
* STORAGE *
*****

*****
```

```

* STATIC STORAGE *
*****
STATIC 121035E8      LENGTH 000003B4 (DECIMAL 948)
ADDRESS OFFSET
121035E8 000000      12103CB8 121036AC 12103CB8 121036C4      1228A6F8 1228A748 1228A800 122
8A848 *.....D..w8..x...y...y.*
12103608 000020      1228A8A8 1228A540 1228A1E8 1228A2C0      00000000 00008000 00000002 000
50002 *..yy..v ..~Y..s{.....*
12103628 000040      00060002 00000000 00000000 00000000      00000000 00000000 02060000 000
00004 *.....*
12103648 000060      02020240 00000005 02040240 00000005      0206FF00 00000004 02040240 000
00007 *.....*
12103668 000080      02020240 0000004C 00000000 00030002      00000003 00000005 00800080 000
00000 *...<.....*
12103688 0000A0      00000001 0C008400 00000050 00000000      00000001 00000001 00000008 000
00001 *.....d....&.....*
121036A8 0000C0      00010000 12289D7C 00000801 00600602      12102FD8 12103680 121036AC 122
8A1AC *.....@.....Q.....~.*
121036C8 0000E0      00444042 00A3AE01 12102FE0 00000000      00000000 C5D9D9D6 D940C3D6 D5C
4C9E3 *.. .t....\.....ERROR CONDIT*
121036E8 000100      C9D6D540 D9C1C9E2 C5C44040 40000000      0000000C 0000000A 00000005 000
00001 *ION RAISED .....*
12103708 000120      00000002 00000005 00000001 D1C1D5C6      C5C2D4C1 D9C1D7D9 D4C1E8D1 E4D
5D1E4 *.....JANFEBMARAPRMAYJUNJU*
12103728 000140      D3C1E4C7 E2C5D7D6 C3E3D5D6 E5C4C5C3      D0000328 00100000 6E3BFFE0 000
00000 *LAUGSEP OCTNOV DEC}.....>..\....*
12103748 000160      00000000 00000000 00000000 90010000      12102FEC 00000000 00000000 000
00000 *.....*
12103768 000180      00000000 00000000 00000000 00200002      00000020 00000026 00000026 000
00001 *.....*
12103788 0001A0      00000001 001F8002 00000041 00000026      00000026 00000001 00000001 000
00043 *.....*
121037A8 0001C0      00000026 00000026 00000001 00000001      00000000 000A0002 0000000A 000
0000D *.....*
121037C8 0001E0      0000000F 00010000 0000000F 00020001      00000010 00000009 00000009 000
00003 *.....*
121037E8 000200      00000001 00000013 00000009 00000009      00000003 00000001 00060002 000
0002B *.....*
12103808 000220      00000004 00000003 00000002 00000001      00000001 00000003 00000001 000
10002 *.....*
12103828 000240      00000000 000A0002 0000000A 0000000D      0000000F 00010000 0000000F 000
20001 *.....*
12103848 000260      00000010 00000008 00000008 00000001      00000001 00000013 00000008 000
00008 *.....*
12103868 000280      00000001 00000001 00000017 00000002      00000001 00000001 00000001 000
00001 *.....*
12103888 0002A0      00000001 00000001 00010002 00000018      00040002 00000000 00000000 000
A0002 *.....*
121038A8 0002C0      0000000A 0000000D 0000000F 00010000      0000000F 00020001 00000010 000
00008 *.....*
121038C8 0002E0      00000008 00000001 00000001 00000013      00000008 00000008 00000001 000
00001 *.....*
121038E8 000300      00000017 00000002 00000001 00000001      00000001 00000001 00000001 000
00001 *.....*
12103908 000320      00010002 00000018 00040002 00000000      00000000 000A0002 0000000A 000
0000D *.....*
12103928 000340      0000000F 00010000 0000000F 00020001      00000010 0000000F 0000000F 000
00003 *.....*
12103948 000360      00000001 00000013 00000015 0000000F      00000003 00000001 00000006 000
00002 *.....*
12103968 000380      00000001 00060002 0000003D 00000004      00000003 00000002 00000001 000
00001 *.....*
12103988 0003A0      00000003 00000001 00010002 00000043      00040002
*.....*

```

External Storage

Each external storage definition is displayed separately as shown in the following screen:

```

*****
* EXTERNAL STORAGE *
*****
    [A]          [B]          [C]
EXTETA      121039A0      LENGTH 00000015 (DECIMAL 21)
ADDRESS      OFFSET
121039A0      000000 [D] C5E7E3C5 D9D5C1D3 0000001C 00000000 00000000 00
    *EXTERNAL.....*

EXTETAX      121039B8      LENGTH 00000015 (DECIMAL 21)
ADDRESS      OFFSET
121039B8      000000 00C5E7E3 C5D9D5C1 D300002C 00000000 00000000 00
    *.EXTERNAL.....*

CNTLAR2      121039D0      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039D0      000000 1228A728
    *..X.*

CNTLAR4      121039D8      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039D8      000000 1228A7C0
    *..X{*

CNTLTAX      121039E0      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039E0      000000 1228A630
    *..W.*

CNTLPTR      121039E8      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039E8      000000 1228A248
    *..S.*

ONE_VAR      121039F0      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039F0      000000 00000000
    *....*

ANOTVAR      121039F8      LENGTH 00000004 (DECIMAL 4)
ADDRESS      OFFSET
121039F8      000000 00000000
    *....*

```

The external storage portion of the report displays the following information:

- **[A]**
Displays the name of the control section created by the PL/I compiler for the external storage area.
- **[B]**
Displays the absolute address of the external storage area.
- **[C]**
Displays the length of the external storage.
- **[D]**
If the external storage display requires more than one line, each line displays the absolute address and the offset within the external storage for the data on that line.



Note: External storage is not displayed for programs that have been compiled using Enterprise PL/I for z/OS and OS/390 with the RENT option.

Program Storage Dump

At the end of the formatted displays for each active PL/I program, the program storage is displayed in dump format.

```
*****
* CSECT  PLITEST1 *
*****
```

```
ADDRESS  OFFSET
121008C0 000000 F2F0F0F4 F0F4F2F0 F1F1F4F9 F5F2F0F3 F0F2F0F0 00280801 02035B00 079
E0000 *20040420114952030200.....$.***
121008E0 000020 04741F07 1F1F0F00 29612102 018C126C 36000301 0FE81210 35E80000 03B
40000 *...../.....%.***Y...Y...
12100900 000040 47F0F022 01C3C5C5 000006B0 000029B0 47F0F001 58F0C31C 184E05EF 000
00000 *.00..CEE.....00..0C..+.....
12100920 000060 07F390EB D00C58E0 D04C4100 E6B05500 C3144130 F03A4720 F01458F0 C28
090F0 *.3..}.|.}<..W...C...0...0B..0*
12100940 000080 E0489210 E00050D0 E00418DE 58503FB2 58603FB6 58103F0A 5010D328 411
00000 *|.k.|.}&|.}|...&...-.....&.L....
12100960 0000A0 5010D32C 58203F0E 5020D330 5010D33C 58103F12 5010D340 41105008 501
050DC *&.L.....&.L.&.L.....&.L..&.&.*
12100980 0000C0 D21FD300 60429200 D2FC9200 D2F81B11 4310D2F4 54103F16 4210D2F4 1B1
14310 *K.L.-.k.K.k.K8....K4.....K4....
121009A0 0000E0 D2F45410 3F1A4210 D2F41B11 4310D2F4 54103F1E 4210D2F4 1B114310 D2F
45410 *K4.....K4....K4.....K4....
121009C0 000100 3F224210 D2F41B11 4310D2F4 54103F26 4210D2F4 1B114310 D2F45410 3F2
A4210 *...K4....K4.....K4....K4.....
121009E0 000120 D2F41B11 4310D2F4 54103F2E 4210D2F4 1B114310 D2F45410 3F324210 D2F
41B11 *K4....K4.....K4....K4.....K4....
12100A00 000140 4310D2F0 54103F16 4210D2F0 1B114310 D2F05410 3F1A4210 D2F01B11 431
0D2F0 *.K0.....K0....K0.....K0....K0*
12100A20 000160 54103F1E 4210D2F0 1B114310 D2F05410 3F224210 D2F01B11 4310D2F0 541
03F26 *.K0.....K0....K0.....K0....K0....
12100A40 000180 4210D2F0 1B114310 D2F05410 3F2A4210 D2F04110 D2AC5010 D5049240 100
0D23F *.K0....K0.....K0..K.&.N.k..K.*
12100A60 0001A0 10011000 5810D504 41106020 58201000 4110D260 50201000 D206D244 606
292C6 *.N.....K-&..K.K.-.kF*
12100A80 0001C0 D24B9286 D24C4140 00041814 5010D4EC 1A114120 D24F4180 D24F41A0 606
9A71E *K.kfK<..&.M.....K|.K|.x.*
12100AA0 0001E0 000841B0 0008189B 47B03174 18910E8A 5820D4EC 89200010 8A200010 411
0D24D *.j....M.i.....K(*
12100AC0 000200 40201000 D208D228 6071D204 D231607A 4110D238 D206D238 607F4110 D23
64120 *...K.K.-.K.K.-:..K.K.K.-".K...
12100AE0 000220 00074020 1000D202 D21C6086 1B114310 D21F5410 3F365610 3F3A4210 D21
F1B11 *.K.K.-f....K.....K...
12100B00 000240 4310D21F 54103F3E 56103F42 4210D21F 1B114310 D21F5610 3F464210 D21
F4110 *.K.....K....K.....K...
12100B20 000260 D21F5010 D4F01B22 43201000 54203F32 56203F4A 42201000 5810D4F0 1B2
24320 *K.&.M0.....M0....
12100B40 000280 10015420 3F4E5620 3F524220 10014110 D2269254 D2264110 D2244120 000
64020 *.....+.....K.k.K..K....
12100B60 0002A0 1000D206 D1D06089 D204D1D7 6090D205 D1DC6095 D202D1E2 609B4110 602
45820 *.K.J}-iK.JP-.K.J.-nK.JS-...
12100B80 0002C0 10004110 D1E55020 1000D204 D1E9609E D204D1EE 609ED204 D1F3609E D20
8D1F8 *...JV&..K.JZ-.K.J.-.K.J3-.K.J8*
12100BA0 0002E0 60A3D209 D20160AC D204D20B 60B64110 60285820 10004110 D2105020 100
0D207 *-tK.K.-.K.K.-...K.&..K.*
12100BC0 000300 D21460BB D205D147 60C39201 D14D9203 D14E9205 D14F9209 D1509211 D15
19221 *K.-.K.J.-Ck.J(k.J+k.J|k.J&k.J.k.*
12100BE0 000320 D1529241 D1534110 00814010 D1544110 01014010 D1564110 02014010 D15
84110 *J.k.J...a..J.....J.....J...
12100C00 000340 04014010 D15A4110 08014010 D15C4110 0FFF4010 D15EA718 20014010 D16
```



```

0A718  *...J!....J*....J;x...J-x,*
12100C20 000360 40014010 D1625810 3F565010 D1645810 3F5A5010 D1685810 3F5E5010 D16
C5810  *...J....&J....!&J....;&J%..*
12100C40 000380 3F625010 D1705810 3F665010 D1745810 3F6A5010 D1785810 3F6E5010 D17
C5810  *...&J....&J....!&J....>&J@..*
12100C60 0003A0 3F725010 D1805810 3F765010 D1845810 3F7A5010 D1885810 3F7E5010 D18
C5810  *...&J....&Jd...:&Jh...=&J...*
12100C80 0003C0 3F825010 D1905810 3F865010 D1945810 3F8A5010 D1985810 3F8E5010 D19
C5810  *...b&J....f&Jm....&Jq....&J...*
12100CA0 0003E0 3F925010 D1A0A718 134A4010 D1A45810 3F965010 D1A84110 003A4010 D1A
CA718  *.k&J.x..¢.Ju...o&Jy....J.x.*

```

The program storage dump can be suppressed by program name or prefix using the CAOUXMOD macro in your installation defaults member, CAOUDFRX. For more information, see [CA SymDump Batch Options \(https://docops.ca.com/display/CAITSD11/CA+SymDump+Batch+Options\)](https://docops.ca.com/display/CAITSD11/CA+SymDump+Batch+Options).

Assembler Reports

This section describes the information provided for Assembler programs.

Program Statistics

For each assembler program in the report, this report contains statistics about the assemble, link, and symbolic postprocessing used when the program was built.

```

*****
* PROGRAM CARXDEMA *
*****
[A]
ASSEMBLED ON 26 JUN,2007 AT 13.25.00
[B]
SYMBOLIC INFORMATION RETRIEVED FROM AD1DEV.USER001.PROTSYM
MEMBER: CARXDEMA DATE: 26 JUN,2007 TIME: 13.25.00 TYPE: PROTSYM
[C]
LINKED ON 26 JUN 2007 AT 13:25:22
LOAD LIBRARY: USER02.LOAD
MODULE LENGTH: 00000958 (DECIMAL 2,392)
LINK OPTIONS: AC(0), AMODE(31), NOOVLY, RENT, REUS, RMODE(24)

```

The program statistics portion of the assembler reports displays the following information:

- **[A]**
Displays the data and time the program was assembled if the program is Language Environment enabled.
- **[B]**
When symbolic information is used to format the report, this section displays the name of the symbolic file and the date and time of the symbolic listing used.
- **[C]**
Displays link-edit information including the date and time that the load module was created, the name of the load library from which the program was loaded, the length of the module, and the linkage editor options used.

Program Status Word

For the abending Assembler program, the Program Status Word (PSW) is displayed as follows:

```

*****
* PROGRAM STATUS WORD *
*****

```

```

[A]
078D3000 80000000 00000000 12100E58
[B]          [C]          [D]
31-BIT ADDRESSING MODE    CC = 3    PRIMARY-SPACE MODE

```

The PSW portion displays the following information.

- **[A]**
Displays the dump-formatted display of the 128-bit PSW.
- **[B]**
Displays the addressing mode (24, 31 or 64) at the time of the ABEND.
- **[C]**
Displays the value of the condition code at the time of the ABEND.
- **[D]**
Displays the address space mode at the time of the ABEND (Primary-Space, Secondary-Space, Home-Space, or Access-Register).

Registers

For each active Assembler program, the values of the general purpose registers at the time when the program was last in control are displayed.

For abending programs, these are the current register values, and the access registers and floating-point and vector registers are also included. For all other programs, these are the registers when the program last transferred control to another program.

The following sample report shows the registers for an abending Assembler program:

```

*****
* REGISTERS AT ABEND *
*****

```

GENERAL REGISTERS:

```

R0  00000000_00000950
R1  00000000_007D06B0
R2  00000000_00006F60
R3  00000000_00000014
R4  00000001_0000001E
R5  00000000_007F2300
R6  00000000_007C4FE0
R7  00000000_7D000000
R8  00000000_007F9780
R9  00000000_007E12F8
R10 00000000_00000000
R11 00000000_007F9260
R12 00000000_12100D28    DISP +000000 IN CSECT CARXDEMA
R13 00000000_00006E70
R14 00000000_12100E8F    DISP +000167 IN CSECT CARXDEMA
R15 00000000_00000000

```

ACCESS REGISTERS:

```

R0 - R3      00000000 00000000 00000000 40C1D9F3
R4 - R7      00000000 40C1D9F5 00000000 40C1D9F7
R8 - R11     00000000 40C1D9F9 00000000 00000000
R12 - R15    00000000 00000000 00000000 00000000

```

FLOATING-POINT REGISTERS:

FPC 00000000					
FR0 00000000 00000000 0.0			FR8 00000000 00000000 0.0		
FR1 00000000 00000000 0.0			FR9 00000000 00000000 0.0		
FR2 00000000 00000000 0.0			FR10 00000000 00000000 0.0		
FR3 00000000 00000000 0.0			FR11 00000000 00000000 0.0		
FR4 00000000 00000000 0.0			FR12 00000000 00000000 0.0		
FR5 00000000 00000000 0.0			FR13 00000000 00000000 0.0		
FR6 00000000 00000000 0.0			FR14 00000000 00000000 0.0		
FR7 00000000 00000000 0.0			FR15 00000000 00000000 0.0		

VECTOR REGISTERS:

VR0 00000000 00000000 00000000 00000000	VR16 00000000 00000000 00000000 00000000
VR1 00000000 00000000 00000000 00000000	VR17 00000000 00000000 00000000 00000000
VR2 00000000 00000000 00000000 00000000	VR18 00000000 00000000 00000000 00000000
VR3 00000000 00000000 00000000 00000000	VR19 00000000 00000000 00000000 00000000
VR4 00000000 00000000 00000000 00000000	VR20 00000000 00000000 00000000 00000000
VR5 00000000 00000000 00000000 00000000	VR21 00000000 00000000 00000000 00000000
VR6 00000000 00000000 00000000 00000000	VR22 00000000 00000000 00000000 00000000
VR7 00000000 00000000 00000000 00000000	VR23 00000000 00000000 00000000 00000000
VR8 00000000 00000000 00000000 00000000	VR24 00000000 00000000 00000000 00000000
VR9 00000000 00000000 00000000 00000000	VR25 00000000 00000000 00000000 00000000
VR10 00000000 00000000 00000000 00000000	VR26 00000000 00000000 00000000 00000000
VR11 00000000 00000000 00000000 00000000	VR27 00000000 00000000 00000000 00000000
VR12 00000000 00000000 00000000 00000000	VR28 00000000 00000000 00000000 00000000
VR13 00000000 00000000 00000000 00000000	VR29 00000000 00000000 00000000 00000000
VR14 00000000 00000000 00000000 00000000	VR30 00000000 00000000 00000000 00000000
VR15 00000000 00000000 00000000 00000000	VR31 00000000 00000000 00000000 00000000

Addressable Storage

For each register that contains a valid storage address, the data immediately preceding and following the storage address is also displayed.

The amount of storage displayed for each register is controlled by the REGMAX installation option. For more information about the REGMAX option, see [Installing \(https://docops.ca.com/display/CAITSD11/Installing\)](https://docops.ca.com/display/CAITSD11/Installing).

In most cases, the addressable storage is displayed in dump format as follows:

[A]				[F]			
R1 = 00000000_007D06B0				(NO ACTIVE USING)			
[B]	[C]	[D]				[E]	
00000000_007D0670	-00040	00000000	00000000	00000000	00000000	*.....*	
00000000_007D0680	-00030	00000000	00000000	00000000	00000000	*.....*	
00000000_007D0690	-00020	00000000	00000000	00000000	00000000	*.....*	
00000000_007D06A0	-00010	00000000	00000000	00000000	00000000	*.....*	
00000000_007D06B0		00000000	00000000	00000000	00000000	*.....*	
00000000_007D06C0	+00010	00000000	00000000	00000000	00000000	*.....*	
00000000_007D06D0	+00020	00000000	00000000	00000000	00000000	*.....*	
00000000_007D06E0	+00030	00000000	00000000	00000000	00000000	*.....*	
00000000_007D06F0	+00040	00000000	00000000	00000000	00000000	*.....*	
00000000_007D0700	+00050	007FBA48	80E03D6A	81509B30	00000094	*"...\. a&....m*	
00000000_007D0710	+00060	007D0784	00000000	7F609000	80E03944	*'.d...."-...\.*	
00000000_007D0720	+00070	00000C00	00000C58	00000C80	007D06F8	*.....'.8*	
00000000_007D0730	+00080	80E0327A	00000004	00FC9980	007D0740	*\.:.....r..'.*	
00000000_007D0740	+00090	00000000	007D0740	80E03542	00000000	*.....'.\.....*	
00000000_007D0750	+000A0	00000000	007D0BF4	00000002	0000FF02	*.....'.4.....*	
00000000_007D0760	+000B0	007FE0A8	007D0B64	0000015C	7F609000	*"...y.'.....*"-...*	
00000000_007D0770	+000C0	007D06F8	80E0327A	00000001	00F50280	*'.8.\.:.....5..*	
00000000_007D0780	+000D0	00000017	007D0740	80FB40D0	00000002	*.....'. .. }....*	
00000000_007D0790	+000E0	00000000	7F609000	7FFA1AE0	00000000	*....."-..."\....*	

```
00000000_007D07A0 +000F0 00000000 0104111C 00000000 00000000 *.
*
```

The addressable storage portion displays the following information:

- **[A]**
Displays the contents of the 64-bit general purpose register containing the valid storage address.
- **[B]**
Displays the absolute address corresponding to the start of the data on the line for each line of the display.
- **[C]**
Displays the offset from the register address corresponding to the start of the data on the line for each line of the display.
- **[D]**
Displays the addressable data in hexadecimal dump format.
- **[E]**
Displays the addressable data converted to EBCDIC display format.
- **[F]**
If symbolic support is available, this indicates whether an active USING was in effect for this register on the abending statement.

If symbolic support is available for the program and the general register has an active USING in effect on the abending statement, the storage may be displayed in merged format. The MRGDSECT option controls whether the merged format should be used when possible.

The following example shows a merged format display:

[A] R13 = 00000000_36013030				[H] DSECT: CEEDSA	
[B]	[C]	[F]	[G]	[D]	[E]
OFFSET	FIELD	NAME	LENGTH	DEF	VALUE
-----	-----	-----	-----	-----	-----
000000	CEEDSAFLGS				0000
*			2	X	
000002	CEEDSAMEMD				0000
*			2	X	
000004	CEEDSABKC				3600E318
*			4	A	
000008	CEEDSAFWC				36013158
*			4	A	
00000C	CEEDSAR14				B600076A
↑*			4	F	
000010	CEEDSAR15				88EB4BB0
*			4	F	
000014	CEEDSAR0				36013158
*			4	F	
000018	CEEDSAR1				3601313C
*			4	F	
00001C	CEEDSAR2				3600C748
*			4	F	
000020	CEEDSAR3				00000014
*			4	F	
000024	CEEDSAR4				007B0064
*			4	F	

```

000028 CEEDSAR5          00000000          *....
*                                4          F
00002C CEEDSAR6          00000000          *....
*                                4          F
000030 CEEDSAR7          00000000          *....
*                                4          F
000034 CEEDSAR8          007F9448          *."m.
*                                4          F
000038 CEEDSAR9          007FF7D0          *."7}
*                                4          F
00003C CEEDSAR10         00000000          *....
*                                4          F
000040 CEEDSAR11         360006A8          *...
y*                                4          F
000044 CEEDSAR12         3600D978          *...R.
*                                4          F
000048 CEEDSALWS         00000000          *....
*                                4          A
00004C CEEDSANAB         36013158          *....
*                                4          A
000050 CEEDSAPNAB        00000000          *....
*                                4          A
000054 .                  00000000 00000000 00000000 00000000 *.....
.....*          16          F
000064 CEEDSATRAN        00          *
*                                1          A
000064 CEEDSARENT        00000000          *....
*                                4          A
000068 CEEDSACILC        00000000          *....
*                                4          A
00006C CEEDSAMODE        00000000          *....
*                                4          A
000070 .                  00000000 00000000          *.....
*                                8          F
000078 CEEDSARMR         00000000          *....
*                                4          A
00007C .                  00000000          *....
*                                4          F
000080 CEEDSAAUTO        00          *
*                                1          D
000080 CEEDSAEND         00000000 00000000          *.....
*                                8          D

```

The merged format portion displays the following information:

- **[A]**
Displays the contents of the 64-bit general purpose register containing a valid storage address with an active USING at the abending statement.
- **[B]**
Displays the offset within the mapping structure corresponding to the line of display.
- **[C]**
Displays the name of the field, if one exists, corresponding to the data displayed on each line.
- **[D]**
Displays the addressable data in hexadecimal dump format.
- **[E]**
Displays the addressable data converted to EBCDIC display format.
- **[F]**
Displays the length of the field being displayed.

- **[G]**
Displays the type of the field as defined in the program.
- **[H]**
Displays the name of the DSECT used to map the addressable storage in this display.

Program Storage Dump

At the end of the formatted displays for each active Assembler program, the program storage for the control section is displayed in dump format.

* CSECT CARXDEMA *

ADDRESS OFFSET

```

12100D28 000000 47F0F070 C3C1D9E7 C4C5D4C1 F0F461F2 F061F0F4 40F1F14B F2F940C1 D3D
3C6E4 *.00.CARXDEMA04/20/04 11.29 ALLFU*
12100D48 000020 E2C9D6D5 40C3C160 D6D7E3C9 D4C9E9C5 D961C9C9 40C3D6D7 E8D9C9C7 C8E
3404D *SION CA-OPTIMIZER/II COPYRIGHT (*
12100D68 000040 C35D40F2 F0F0F440 C3D6D4D7 E4E3C5D9 40C1E2E2 D6C3C9C1 E3C5E240 C9D
5E3C5 *(C) 2004 COMPUTER ASSOCIATES INTE*
12100D88 000060 D9D5C1E3 C9D6D5C1 D36B40C9 D5C34B40 90ECD00C 18CF4130 00F01803 451
0C080 *RNATIONAL, INC. ...}.0....{*
12100DA8 000080 0A0A1821 18411F51 0E2450D0 10045010 D00818D1 41F00004 4110C2C8 584
0021C *.0....&}.&}.J.0....BH. ...*
12100DC8 0000A0 5840400C 41404018 17334330 40001233 4780C0C8 D5074004 10004780 C0C
64143 *. ... {HN. ....{F.*
12100DE8 0000C0 400047F0 C0A817FF 12FF4770 C1C29A0F C214E340 C2BC0004 4DF0C0E4 C3C
1D6C4 *.0{y....AB..B.T B...{UCA0D*
12100E08 0000E0 E2D5C1D7 4100F000 1B110A08 18F005EF 4500C0FC C3C1D6C4 E2D5C1D7 0A0
9D25F *SNAP..0....0....{CA0DSNAP..K^*
12100E28 000100 D074C254 D203D0D4 C2B44110 D0D494F0 1000960F 100043E1 00004100 D07
45001 *.B.K.}MB...}Mm0..o.....}.&.*
12100E48 000120 000042E1 00000A13 9110D0A4 47E0C1EC FA30D050 C2D0D203 D0D8C2B8 411
0D0D8 *. ...j.}u.\A...}&B}K.}QB...}Q*
12100E68 000140 43E10000 4100D074 50010000 42E10000 0A144110 D0749101 10174710 C1A
01FFF *. ...}.&.....}.j....A...*
12100E88 000160 BFF71015 58E0F000 12EE4770 C1769130 F0044770 C1A09601 10171BEE 1B1
1BF13 *.7...\0....A.j.0...A.o.....*
12100EA8 000180 F00643E0 F0051C0E 41101008 9140F004 47E0C198 41101008 18014110 F00
0A0A0 *0...\0.....j 0..\Aq.....0...*
12100EC8 0001A0 5820D04C 181D58D0 D0044100 00F01800 41101000 0A0A58E0 D00C18F2 980
CD014 *.<...}}....0.....\}..2q.}*
12100EE8 0001C0 0B0E0700 4510C1E6 00198000 E2E8E2D7 D9C9D5E3 40C4C440 D5D6E340 C6D
6E4D5 *. ....AW....SYSPRINT DD NOT FOUN*
12100F08 0001E0 C4000000 20000A23 47F0C130 4510C20E 001A8000 C5D9D9D6 D940D6D7 C5D
5C9D5 *D.....0A...B....ERROR OPENIN*
12100F28 000200 C740E2E8 E2D7D9C9 D5E30000 00200A23 47F0C130 40C1D9F0 40C1D9F1 000
00000 *G SYSPRINT.....0A. AR0 AR1...*
12100F48 000220 40C1D9F3 00000000 40C1D9F5 00000000 40C1D9F7 00000000 40C1D9F9 000
00000 * AR3.... AR5.... AR7.... AR9...*
12100F68 000240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000
00000 *. ...*
12100F88 000260 00000000 00000000 00000001 00004000 00000001 00000001 90000000 E2E
8E2D7 *. ...SYSP*
12100FA8 000280 D9C9D5E3 02000050 00000001 00000001 00000000 00000000 00000001 000
00001 *RINT...&.....*
12100FC8 0002A0 00000001 00000050 00000001 00000000 00000001 80100FDD 80100FE1 000
00001 *. ...&.....*
12100FE8 0002C0 0000001E 00000000 E2E8E2D7 D9C9D5E3 1C
*. ....SYSPRINT.*

```

The program storage dump can be suppressed by program name or prefix using the CAOUXMOD macro in your installation defaults member, CAUDFRX. For more information about the CAOUXMOD macro see [CA SymDump Batch Options \(https://docops.ca.com/display/CAITSD11/CA+SymDump+Batch+Options\)](https://docops.ca.com/display/CAITSD11/CA+SymDump+Batch+Options).

Open Files Report

In the Open Files report, CA SymDump Batch formats information from the data management control blocks and dumps the current logical record for all files in the job step that remain open and are not displayed earlier in the File Section of the abend report. If the file was accessed sequentially, the contents of the previous record may also be displayed. Optional sections of the report include catalog information for VSAM files and dumps of data management control blocks. The contents of this report are similar to the contents of the File Section Display.

The Open Files report appears once near the end of the abend report. The reported files may be from COBOL programs not reported on in the abend report or from any non-COBOL program in the job step. An example of the Open Files report follows. For a description about the optional sections, see [File Section Display](#).

```
*****
* OPEN FILES REPORT *
*****
```

```
DDNAME=SYSUDUMP  DSORG=PS  BSAM  BUFNO=0
DSN=USER02.USER02A.JOB59924.D0000110.?  RECFM=VBA
STATUS(OPEN)  USAGE(OUTPUT)  LRECL=125  BLKSIZE=1632
```

```
[A] DDNAME=PRINT  DSORG=PS  QSAM  BUFNO=5  UNIT=3390,UCB  VOL=SER=WRKD28
   DSN=USER02.PRINT  RECFM=FB
[B] STATUS(OPEN)  USAGE(OUTPUT)  LRECL=120  BLKSIZE=120
```

[C] CONTENTS OF CURRENT LOGICAL RECORD

```
-----
ADDRESS 22BBBE20  LENGTH 00000078  (DECIMAL 120)
      OFFSET
+000000  40404040  40404040  40404040  40404040  40404040  40404040  40404040  40
404040  *
+000020  40404040  40404040  40404040  40404040  40404040  40404040  40404040  40
404040  *
+000040  40404040  40404040  40404040  40404040  40404040  40404040  40404040  40
404040  *
+000060  40404040  40404040  40404040  40404040  40404040  40404040  40404040
      *
      *
```

[D] CONTENTS OF PREVIOUS RECORD

```
-----
      LENGTH 00000078  (DECIMAL 120)
      OFFSET
+000000  40404040  40404040  40404040  40404040  40404040  40404040  40404040  404
04040  *
+000020  40404040  40404040  40404040  40404040  40404040  40C3C160  E2E8D4C4  E4D
4D761  *
+000040  C2C1E3C3  C8404040  40404040  40404040  40404040  40404040  40404040  404
04040  *
*BATCH
+000060  40404040  40404040  40404040  40404040  40404040  40404040  40404040
      *
      *
```

The Open Files Report displays the following information:

- **[A]**
Displays the data management control blocks. The data management control blocks are interpreted and displayed in KEYWORD=value format. This information includes the ddname, data set organization, access method, number of buffers allocated, and data set name. For non-VSAM files, the unit type, volser and record format are also displayed.
- **[B]**
Indicates the file's current status (OPEN, CLOSED) and usage (INPUT, OUTPUT, etc.), and the record length. For non-VSAM files, the block size is also displayed, while the feedback code, function code, and return code are displayed for VSAM files.
- **[C]**
Displays the contents of the current logical record in hexadecimal format, with a character representation on the right. If the FIRST128 option was specified, only the first 128 bytes of the record are displayed.
For keyed VSAM files, the current record key will be displayed in the same format, prior to the contents of the record. (See [File Section Display](#) for an example.)
- **[D]**
Displays the contents of the previous record data for sequentially accessed files (VSAM, QSAM, and BSAM), if it is available in an I/O buffer and:
 - The file is not a BSAM file using NOTE/POINT
 - The file is not a VSAM file using skip-sequential access
 - The file is not a JES managed data set, such as in-stream data or data written to a SYSOUT class
 - The file does not contain unformatted, spanned, compressed, or extended records

For keyed VSAM files, the previous record key will be displayed prior to the contents of the previous record. (For an example, see [File Section Display .](#))

Save Area Trace Report

The Save Area Trace report assists you in tracing and debugging programs in a multiple-module environment. This report prints only once for any given ABEND reporting sequence. Whether the ABEND occurs in a multiple-module run and CA SymDump Batch reports on more than one program, or it is a single-module run with only the abend report, the Save Area Trace always prints just once.

When a program calls a subordinate program, the caller's registers are stored in an eighteen-fullword save area to establish the means of return. A series of calling and called programs establishes a chain of save areas, as shown in the following example.

```
*****
* SAVE AREA TRACE *
*****
```

SAVE AREA FOR SYSTEM

[A]	[B]	[C]	[D]	[E]
SA 00006F60	WD1 00000000	BKL 00000000	FWL 00000000	RET 80FE0708
F	EPA B6500888	G R0 FD000008	R1 00006FF8	R2 00000040
	R3 007D9D84	R4 007D9D60	R5 007FF5E0	R6 007B7FE0
	R7 FD000000	R8 007F8088	R9 007FF200	R10 00000000
	R11 007FF5E0	R12 80C999FA		

SYSTEM CALLED CEE (ASSEMBLER)
SAVE AREA FOR CEE

SA 365113C0	WD1 00000000	BKL 00006F60	FWL 00000000	RET 00CA5AB0
	EPA 36500888	R0 00000000	R1 3650F7B8	R2 00000000
	R3 00000000	R4 00000000	R5 00000000	R6 00000000
	R7 00000000	R8 00000000	R9 00000000	R10 00000000
	R11 00000000	R12 36510A20		

CEE CALLED CAOEDEMO (COBOL FOR Z/OS NOT CA-OPT)

CAOEDEMO ABENDED

In this example, the program called by the system is CAOEDEMO. (The intervening save area is for the LE runtime).

The following are the contents of the save area (all addresses are absolute hexadecimal):

- **[A]**
Displays the system's save area location.
- **[B]**
Displays the first word of the eighteen-word save area. It is not used by the system and is usually zeros.
- **[C]**
Points backwards to the location of the prior save area (if any). BKL stands for backward link.
- **[D]**
Points forward to the location of the save area belonging to the next program. FWL is the forward link.
- **[E]**
Represents the return address (RET) as an absolute hexadecimal location.
- **[F]**
Represents the point where CAOEDEMO is entered when control is passed from the System to CAOEDEMO. EPA means entry point address.
- **[G]**
The next thirteen words, zero through twelve, save the contents of registers.

An eighteen-word save area prints for each active program in the hierarchy leading to the program that abended. The save area for the abending program does not display.

Whenever an unidentifiable program is encountered in the chain, it designates the program in the Save Area Trace as UNKWN nnn , where nnn is 1 to 999. CA SymDump Batch does not determine the program's language.

The information that is supplied in this detailed version of the Save Area Trace can be valuable to the user who understands register conventions and whose problem involves more than one module.

IMS Report

The IMS portion of the abend report is produced whenever a program abends which accesses an IMS database.

```
*****
* IMS 8.1.0 RELATED INFORMATION * [A]
*****
```

```
*****
* LAST IMS CALL * [B]
*****
```

```
IMS FUNCTION: ISRT [C]
```

The IMS report begins with the following information:

- [A]
Displays the release of IMS.
- [B]
If the location of the last IMS call can be determined, it is displayed in the same format as the abending statement information (not shown).



Note: Call location information is not available under the LE run time.

- [C]
Displays the IMS function used on the last IMS call.

Information about the current or last database PCB is then displayed, if it is available. An example follows:

```
*****
* CURRENT/LAST DATABASE PCB - DI21PART * [A]
*****

PCB PREFIX:
ADDRESS  OFFSET  [B]

00088CA8 000000 00500038 00020028 40404040 00000000 00000000 00000000 00000000 C4
C2D3D6  *.&..... DBLO*
00088CC8 000020 C1C44040 000060E0 00000000 00000000 00088D30 00088CA8
               *AD ..-\.....y*
```

PCB: [C]

LINE#	LEVEL/FIELD NAME	VALUE
/LOCATION	DEFINITION	
000096	01 PCB-AREA-	
1	BLL=0001+000000 (00088CE0)	
000097	02 DBD-	
NAME	*DI21PART*	X
(08)		
000098	02 SEGMENT-	
LEVEL	*02*	X

```

(02)
000099 02 STATUS-
CODES                                SPACES                                X
(02)
000100 02 PROCESS-
OPTIONS                             *L  *                                X
(04)
000101 02 FILLER                                +560296
      S9(05) BINARY
000102 02 SEG-
NAME                                *STANINFO*                                X
(08)
  [D]  [E]      [F]      [G]      [H]      [I]
STATUS SEGMENT    NUMBER OF    SEGMENT    PROCESSING    KEY
CODE   LEVEL      SEGMENTS    NAME      OPTIONS    LENGTH
' '
02      5          STANINFO    L          19

```

FEEDBACK FROM LAST CALL: [J]

ADDRESS OFFSET

```

00088D04 000000  F0F2F9F8 F9F0F3F6 60F0F0F1 40404040  40F0F2
      *02989036-001  02*

```

TRACE OF DATABASE CALLS (NEWEST ENTRY FIRST):

```

-0 ISRT      OK      [K]
-1 ISRT      OK
-2 ISRT      OK
-3 ISRT      OK
-4 ISRT      OK
-5 ISRT      OK

```

The IMS report displays the following information about the current or last database PCB:

- **[A]**
Displays the PCB name.
- **[B]**
Displays the address and contents of the PCB prefix (IMS 4.1.0 and above).
- **[C]**
Displays the address and contents of the PCB. The PCB is displayed in merged format if all of these conditions are met:
 - The MERGEDB option is ON.
 - The PCB is defined in the most active COBOL, PL/I, or Assembler program.
 - Symbolic information is available for that program.

Otherwise, the PCB is displayed in dump format.

- **[D]**
Displays the status code.
- **[E]**
Displays the segment level.
- **[F]**
Displays the number of segments.
- **[G]**
Displays the name of the last retrieved segment.
- **[H]**
Displays the processing options.
- **[I]**
Displays the key length.
- **[J]**
Displays the key feedback.
- **[K]**
Displays the call trace information for the JCB.

The rest of the information about the last IMS call is displayed as follows:

* CURRENT/LAST SEGMENT * **[A]**

LINE# /LOCATION	LEVEL/FIELD NAME	DEFINITION	VALUE
000081 AREA	01 SEG00060-INSERT-	BLW=0000+0004A8 (3667F578)	
000082 X(61)	02 FILLER		*02 742 *
		(+000032) *	1200 96 *
000083 (03)	02 RIGHT-MAKE-SPAN		+63 S9
000084 (06)	02 FILLER		SPACES X
000085 (03)	02 WRONG-MAKE-SPAN	?	*06C* 9
000086 (12)	02 FILLER		SPACES X

* SSA FOR LAST CALL * **[B]**

237/326

LINE# /LOCATION	LEVEL/FIELD NAME	DEFINITION	VALUE
000044 SSA	01 SEG00010-	BLW=0000+0003D0	(3667F4A0)
000045 (08)	02 SEG-NAME-00010		*PARTR00T* X
000046 (01)	02 BEGIN-OP-00010		*(* X
000047 (08)	02 KEY-NAME-00010		*PARTKEY * X
000048 (02)	02 REL-OPER-00010		* =* X
000049 (17)	02 KEY-VALUE-00010		*02989036-001 * X
000050 (01)	02 END-OP-00010		*)* X

The IMS report displays the following information about the current/last segment and the Segment Search Argument (SSA) for last call:

- **[A]**
Displays the current or last segment returned in the user I/O area.
- **[B]**
Displays the SSA for the last database call, if available. If multiple search arguments were passed on the last call, they are displayed in the order in which they were passed.

The segment and SSAs are displayed in merged format if all of the following conditions are met:

- The MERGEDB option is ON.
- The area is defined in the most active COBOL, PL/I, or Assembler program.
- Symbolic information is available for that program.

Otherwise, they are displayed in dump format.

All of the program communication blocks (PCBs) used by the program are displayed, beginning with the terminal PCBs, as shown next:

```

*****
* PROGRAM COMMUNICATION BLOCKS *
*****

[A]                                     [B]

*****
* TERMINAL PCB FOR          - RELATIVE NUMBER 1      * (CURRENT OR LAST USED)
*****

PCB PREFIX:  [C]
ADDRESS  OFFSET
00049084 000000    00400038 00010018 40404040 00000000    00000000 00000000 00000000 C9
D6D7C3  *.....          .....IOPC      *
000490A4 000020    C2404040 00000000 00000000 40404040    40404040 00049084
*B          ....d*

```

```

PCB:      [D]
ADDRESS  OFFSET
000490BC 000000  40404040 40404040 10004040 40404040  40404040 40404040 40404040 40
404040  *
000490DC 000020  40404040 40404040 40404040 40404040  00000000 00000000 00000000 00
000000  *
[E]      [F]      [G]      [H]
STATUS    DATE AND TIME OF  MESSAGE    FORMATTING
CODE      LAST MESSAGE      COUNT      N/A
' '      N/A                N/A                N/A

```

The Program Communication Blocks portion displays the following information:

- [A]
Displays the PCB name and number.
- [B]
Indicates whether it is the current or last used terminal PCB.
- [C]
Displays the address and contents of the PCB prefix (IMS 4.1.0 and above).
- [D]
Displays the address and contents of the PCB. The PCB is displayed in merged format if all of these conditions are met:
 - The MERGEDB option is ON.
 - The PCB is defined in the most active COBOL, PL/I, or Assembler program.
 - Symbolic information is available for that program.

Otherwise, the PCB is displayed in dump format.

- [E]
Displays the status code of the PCB.
- [F]
Displays the date and time of the last message.
- [G]
Displays the message count.
- [H]
Displays the formatting attributes.

The following screen displays the database PCBs:

```

[A]      [B]
*****
* DATABASE PCB FOR DI21PART - RELATIVE NUMBER 2      * (CURRENT OR LAST USED)

```

PCB PREFIX: [C]

ADDRESS OFFSET

```

00088CA8 000000 00500038 00020028 40404040 00000000 00000000 00000000 C4
C2D3D6 *.&.....DBL0*
00088CC8 000020 C1C44040 000060E0 00000000 00000000 00088D30 00088CA8
*AD ..-\.....y*

```

PCB: [D]

LINE# /LOCATION	LEVEL/FIELD NAME	DEFINITION	VALUE		
000096 1	01 PCB-AREA-	BLL=0001+000000 (00088CE0)			
000097 NAME (08)	02 DBD-	*DI21PART*	X		
000098 LEVEL (02)	02 SEGMENT-	*02*	X		
000099 CODES (02)	02 STATUS-	SPACES	X		
000100 OPTIONS (04)	02 PROCESS-	*L *	X		
000101 S9(05)	02 FILLER		+560296		
000102 NAME (08)	02 SEG-	*STANINFO*	X		
[E]	[F]	[G]	[H]	[I]	[J]
STATUS	SEGMENT	NUMBER OF	SEGMENT	PROCESSING	KEY
CODE	LEVEL	SEGMENTS	NAME	OPTIONS	LENGTH
02	5	STANINFO	L	19	

FEEDBACK FROM LAST CALL:

ADDRESS OFFSET [K]

```

00088D04 000000 F0F2F9F8 F9F0F3F6 60F0F0F1 40404040 40F0F2
*02989036-001 02*

```

TRACE OF DATABASE CALLS (NEWEST ENTRY FIRST):

```

-0 ISRT      OK
-1 ISRT      OK [L]
-2 ISRT      OK
-3 ISRT      OK

```

-4 ISRT	OK
-5 ISRT	OK

The Database PCBs portion displays the following information:

- **[A]**
Displays the PCB name and number.
- **[B]**
Indicates whether it is the current or last used database PCB.
- **[C]**
Displays the address and contents of the PCB prefix (IMS 4.1.0 and above).
- **[D]**
Displays the address and contents of the PCB. The PCB is displayed in merged format if all of these conditions are met:
 - The MERGEDB option is ON.
 - The PCB is defined in the most active COBOL, PL/I, or Assembler program.
 - Symbolic information is available for that program.

Otherwise, the PCB is displayed in dump format.

- **[E]**
Displays the status code.
- **[F]**
Displays the segment level.
- **[G]**
Displays the number of segments.
- **[H]**
Displays the name of the last retrieved segment.
- **[I]**
Displays the processing options.
- **[J]**
Displays the key length.
- **[K]**
Displays the key feedback.
- **[L]**
Displays the call trace information for the JCB.

* PARAMETER LIST * **[A]**

241/326

[G] TRANSACTION NAME	NO TRANSACTION NAME
[H] USERID FROM PST	UNAVAILABLE

The Environment Data portion displays the following information:

- [A]
Displays the identifier from the execute parameters.
- [B]
Displays the control region type; BATCH, DB/DC, and so on.
- [C]
Displays the Application region type; BATCH, MPP, BMP, and so on.
- [D]
Displays the region identifier.
- [E]
Displays the name of the application program being run.
- [F]
Displays the name of the PSB currently allocated.
- [G]
Displays the name of the current transaction, if applicable.
- [H]
Displays the user ID from the PST, if available.

DB2 Report

The DB2 portion of the abend report is produced whenever a program that accesses a DB2 database using DB2 for MVS/ESA Release 4.1.0 or above abends. The first page of the DB2 report displays the following information:

```
**** [A] ****
* DB2 8.1.0 RELATED INFORMATION FOR SUBSYSTEM D81A * [B]
****
```

```
*****
* RETURN CODE FROM LAST SQL CALL * [C]
*****
```

```
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
```

```
*****
* LAST SQL STATEMENT *
*****
```

```
STATEMENT LOCATION: [D]
```

```
COLLECTION-ID      OPT2DB2DEMO
PACKAGE NAME       DB2DEMO
CONSISTENCY TOKEN  1841B4DE029F3FE6
SECTION NUMBER     00012
STATEMENT NUMBER   00755
```

```
SQL STATEMENT: [E]
```

```

SELECT EMP , FIRSTNAME , LASTNAME , DEPT , MANAGER INTO :DCLEMPLOY.EMPLOY-EMP
:EMP-IND-1.EMP-IND-EMP , :DCLEMPLOY.EMPLOY-FIRSTNAME :EMP-IND-1.EMP-IND-FNAME ,
:DCLEMPLOY.EMPLOY-LASTNAME :EMP-IND-1.EMP-IND-LNAME , :DCLEMPLOY.EMPLOY-DEPT
:EMP-IND-1.EMP-IND-DEPT , :DCLEMPLOY.EMPLOY-MANAGER :EMP-IND-1.EMP-IND-MANAGER
FROM EMPLOY WHERE EMP = :DCLJOB.JOB-EMP

```

The DB2 and the LAST SQL Statement portions of the abend report display the following information:

- **[A]**
Displays the release of DB2.
- **[B]**
Displays the DB2 subsystem-ID.
- **[C]**
Displays the SQL return code, SQLCODE, and the message associated with that code. If SQLCODE is non-zero, the SQLSTATE value also displays.
- **[D]**
Displays the location of the last SQL statement executed. (This is the last SQL statement before the ABEND or the abending SQL statement.)
If the DB2 program containing the last SQL statement is bound into a package, the location information includes the collection-ID, package name, consistency token, section number, and statement number (as shown in the example).
If the DB2 program is not bound into a package, the location information includes the plan name, DBRM name, consistency token, section number, and statement number.
In both cases, the statement number refers to the listing generated by the SQL precompile, *not* the COBOL compile listing.
- **[E]**
Displays the last or abending SQL statement. If there is an associated cursor declaration, it is also displayed.

If the last SQL statement is not found in the DB2 catalog (for packages or plans which were bound remotely, for example), the statement type displays.

```

*****
* SQL DESCRIPTOR AREA (OUTPUT) *      [A]
*****

```

LINE# /LOCATION	LEVEL/FIELD NAME	VALUE DEFINITION
001225	05 SQL-AVAR-LIST31	BLW=0001+0008B4 (000669EC)
001226	10 PRE-SQLDAID	? X 'E2D8D3C4C1404008' X
(8)		
001227	10 PRE-SQLDABC	+236 S9
(9) BINARY		
001228	10 PRE-SQLN	+5 S9
(4) BINARY		
001229	10 PRE-SQLLD	+5 S9
(4) BINARY		
001230	10 PRE-SQLVAR	BLW=0001+0008C4 (000669FC)
001231	12 SQLVAR-BASE1	BLW=0001+0008C4 (000669FC)

```

001232 15 SQL-AVAR-
TYPE1                                +501                                S9
(4) BINARY
001233 15 SQL-AVAR-LEN1                                +2
.
.
.
*****
* HOST VARIABLES (OUTPUT) *      [B]
*****

LINE#    LEVEL/FIELD NAME                VALUE
/LOCATION                DEFINITION
-----
000311  01 DCLEMPLOY                      BLW=0000+000940  (00065A78)
000312  10 EMPLOY-EMP                      +7                                S9
(4) BINARY
000322  01 EMP-IND-TABLE                    BLW=0000+000988  (00065AC0)
000323  10 EMP-IND-1                      BLW=0000+000988  (00065AC0)
000324  20 EMP-IND-
EMP                                +0                                S9
(4) BINARY
000331  10 EMP-IND-2
(1)                                +0                                S9
(4) BINARY REDEFINES EMP-IND-1

```

The SQL Descriptor Area and the Host Variables portions of the report display the following information:

- **[A]**
Displays the contents of the SQLDAs for the last SQL statement, if applicable. If the MERGEDB option is ON and symbolic information is available for the program, the SQLDA is mapped. Otherwise, the SQLDA is displayed in dump format.
- **[B]**
Displays the contents of the host variables for the last SQL statement, if applicable. If relevant, the output host variables are displayed following the SQLDA for the output variables, and the input host variables are displayed following the SQLDA for the input variables. If the MERGEDB option is ON and symbolic information is available for the program, the host variables are mapped. Otherwise, the fully qualified names of the host variables are displayed, with the address, length, and data in dump format.

If DB2 objects are referenced by the last SQL statement, the names and types of the objects are displayed after the Host Variables. Similarly, if columns in a table or view are referenced, the contents of the referenced columns are displayed.

```

*****
* SQL COMMUNICATIONS AREA *      [A]
*****

```

```

LINE#    LEVEL/FIELD NAME                VALUE
/LOCATION                DEFINITION
-----
000219  01 SQLCA                          BLW=0000+000848  (00065980)

```

```

000220 05 SQLCAID                      *SQLCA  *
      X(8)
000221 05 SQLCABC                      +136
      S9(9) BINARY
000222 05 SQLCODE                      +0
      S9(9) BINARY
000223 05 SQLERRM                      BLW=0000+000858 (00065990)

000224 49 SQLERRML                      +0
      S9(4) BINARY
000225 49 SQLERRMC                      SPACES
      X(70)
000226 05 SQLERRP                      *DSN    *
      X(8)
000227 05 SQLERRD                      +0                      S9
      (1)
      (9) BINARY OCCURS 6
      <UNMERGED DATA FOR SQLERRD> X '0000000000000000FFFFFFFF00000000' LE
NGTH 20
      (+000016) X '00000000'

000229 05 SQLWARN                      BLW=0000+0008C0 (000659F8)

000230 10 SQLWARN0                      SPACES
      X

```

The SQL Communications Area portion of the report displays the following information:

- **[A]**
Displays the contents of the SQLCA. If the MERGEDB option is ON and symbolic information is available for the program, data item information such as the data name, picture clause, and usage type are merged with the SQLCA storage. Otherwise, the address, length, and contents of the SQLCA are displayed in dump format.

The connection information is displayed, as shown in the following example:

```

*****
* CONNECTION INFORMATION *
*****

[A] CONNECTION MODE    BACKGROUND
[B] CONNECTION TYPE    BATCH
[C] CONNECTION ID      BATCH
[D] CORRELATION ID     USER001D
[E] AUTHORIZATION ID   USER001

```

The Connection Information portion of the report displays the following information:

- **[A]**
Displays the connection mode.
- **[B]**
Displays the connection type.
- **[C]**
Displays the connection ID.
- **[D]**
Displays the correlation ID.
- **[E]**
Displays the authorization ID.

The plan information follows the connection information:

```
*****
* APPLICATION PLAN OPT2DEMO * [A]
*****
```

```
BIND DATE/TIME 06 MAR,2008 14.57.54 [B]
BOUND BY      USER001 [C]
```

BIND OPTIONS: [E]

```
ACQUIRE(USE), CACHESIZE(3072), CURRENTDATA(YES), CURRENTSERVER(),
DBPROTOCOL(DRDA), NODEFER(PREPARE), DEGREE(1), DISCONNECT(EXPLICIT),
DYNAMICRULES(RUN), ENCODING CCSID(500), EXPLAIN(NO),
FUNCTIONTS(2008-03-06-14.57.54.294632), GROUP MEMBER(), IMMEDIATEWRITE(NO),
ISOLATION(CS), KEEP DYNAMIC(NO), OPERATIVE(YES), OPTHINT(), OWNER(USER001 ),
PATHSCHEMAS(), QUALIFIER(CAOPTII ), RELBOUND(L), RELEASE(DEALLOCATE),
REOPT(NONE), SQLRULES(DB2), VALID(YES), VALIDATE(RUN)
```

TABLE DEPENDENCIES: [F]

```
CAOPTII.DEPART
CAOPTII.DIVISN
CAOPTII.EMPLOY
CAOPTII.JOB
CAOPTII.TITLES
```

TABLE SPACE DEPENDENCIES: [F]

```
CAOPTII.DEPART
CAOPTII.DIVISN
CAOPTII.EMPLOY
CAOPTII.JOB
```

The Application Plan portion of the report displays the following information:

- [A]
Displays the application plan name.
- [B]
Displays the date and time when the plan was bound.
- [C]
Displays the ID of the user who bound the plan.
- [D]
Displays the comment associated with the plan, if applicable (not shown).
- [E]
Displays the bind options.
- [F]
Displays the database objects on which the plan is dependent.

The package information is displayed for each package in the plan. If the DB2ACTIV option is specified, package information is displayed only for the packages that are active at the time of the ABEND. An example showing the package information follows:

```
*****
```

* PACKAGE SUMMARY FOR PLAN OPT2DEMO *

* PACKAGE DB2ADD * [A]

[B] COLLECTION-
ID OPT2DB2DEMO

[C] BIND DATE/TIME 06 MAR,
2008 14.57.52

[D] PRE-COMPILE DATE/TIME 06 MAR,
2008 14.57.22

[E]
CONSISTENCY TOKEN 1841B4DE1D8BFF84

[F] DBRM LIBRARY USER001.DBRMLIB.
D810

[H]
BIND OPTIONS:

CREATOR(USER001), DBPROTOCOL(DRDA), CURRENTDATA(C), DEGREE
(1),
ENCODING CCSID(500), EXPLAIN(NO), FUNCTIONTS(2008-03-06-
14.57.52.723939),
GROUP MEMBER(), IMMEDIATEWRITE(NO), ISOLATION(CS), KEEP DYNAMIC(NO), OPERATIVE
(YES),
OPTHINT(), OWNER(USER001), PATHSCHEMAS(), QUALIFIER(CAOPTII), RELBOUND
(L),
RELEASE(DEALLOCATE), REMOTE(NO), REOPT(NONE), SQLERROR
(NOPACKAGE),
TYPE(BINDPKG), VALID(YES), VALIDATE
(RUN)

PRE-COMPILE OPTIONS: [I]

APOST, DEC(31), NOGRAPHIC, HOST(IBMCOB), NOTKATAKANA, PERIOD, VERSION()

TABLE DEPENDENCIES: [J]

CAOPTII.
DEPART

CAOPTII.
DIVISN

CAOPTII.
EMPLOY

CAOPTII.
JOB

CAOPTII.
TITLES

TABLE SPACE DEPENDENCIES:

CAOPTII.DIVISN
CAOPTII.EMPLOY
CAOPTII.DEPART
CAOPTII.JOB

The Package Summary portion consists of the following information:

- **[A]**
Displays the package name.
- **[B]**
Displays the collection ID.
- **[C]**
Displays the bind date and time.
- **[D]**
Displays the precompile date and time.
- **[E]**
Displays the consistency token.
- **[F]**
Displays the library where the DBRM resides.
- **[G]**
Displays the comment associated with the package, if applicable (not shown).
- **[H]**
Displays the bind options.
- **[I]**
Displays the precompile options.
- **[J]**
Displays the database objects on which the package is dependent.

Finally, CA SymDump Batch displays the information for DBRMs within the plan not bound into a package. If the DB2ACTIV option is specified, DBRM information is displayed only for the DBRMs that are active at the time of the ABEND.

```
*****
* DBRM SUMMARY FOR PLAN OPT2DEMO * [A]
*****
```



```
*****
* DBRM DB2DEL *
*****
```

```
[B] PRE-COMPILE DATE/TIME 06 MAR,2008 14.57.26
[C] CONSISTENCY TOKEN    1841B4DF0D4A201A
[D] DBRM LIBRARY         USER001.DBRMLIB.D810
```

PRE-COMPILE OPTIONS: [E]

APOST, DEC(31), NOGRAPHIC, HOST(IBMCOB), NOTKATAKANA, PERIOD, VERSION()

The DBRM Summary for Plan portion consists of the following information:

- [A]
Displays the DBRM name.
- [B]
Displays the precompile date and time.
- [C]
Displays the consistency token.
- [D]
Displays the library where the DBRM resides.
- [E]
Displays the precompile options.

CA IDMS/DB Report

The CA IDMS/DB portion of the abend report is produced whenever a program that accesses a CA IDMS/DB database using CA IDMS/DB r12.0 or above abends. The reports are generated based on the SYSIDMS parameters specified. Information about these parameters can be found in the CA IDMS/DB documentation .

The first page of the CA IDMS/DB report follows:

```
*****
* CA-IDMS REPORT *
*****
```

CA-IDMS abend trace Tape Genlevel: G0GJ6M Release: 1600 [A]

User=USER02 [B] Batch Local Job

DBNODE= DBNAME=EMPDEMO DICTNODE= DICTNAME= [C]

```
*****
* SYSIDMS parms * [D]
*****
```

```
ECHO=ON
DMCL=R160DMCL
DBNAME=EMPDEMO
ABENDTRACE=ON ABENDTRACE_ENTRIES=255
ABENDTRACE_VIBSNAP=ON
ABENDTRACE_SUBSCHEMA_DISPLAY=ON
```

* DML trace for subschema=EMPSS01 * [E]

```

VERB=59 BIND SUBSCHEMA--
>EMPSS01 DBNAME=EMPDEMO PROGRAM=CA02IDMS Caller=CA02IDMS DMLSEQ=0000
01 *** I D M S
VERB=48 BIND Record REC--
>STRUCTURE ADDR=000403A0 Caller=CA02IDMS DMLSEQ=000002 *** I D M S
VERB=48 BIND Record REC--
>SKILL ADDR=000403B0 Caller=CA02IDMS DMLSEQ=000003 *** I D M S
VERB=48 BIND Record [F] REC--
>OFFICE ADDR=00040400 Caller=CA02IDMS DMLSEQ=000004 *** I D M S
VERB=48 BIND Record REC-->NON-HOSP-
CLAIM ADDR=00040450 Caller=CA02IDMS DMLSEQ=000005 *** I D M S
VERB=48 BIND Record REC--
>JOB ADDR=00040870 Caller=CA02IDMS DMLSEQ=000006 *** I D M S
VERB=48 BIND Record REC-->INSURANCE-
PLAN ADDR=00040998 Caller=CA02IDMS DMLSEQ=000007 *** I D M S
VERB=48 BIND Record REC-->HOSPITAL-
CLAIM ADDR=00040A20 Caller=CA02IDMS DMLSEQ=000008 *** I D M S
VERB=48 BIND Record REC--
>EXPERTISE ADDR=00040B50 Caller=CA02IDMS DMLSEQ=000009 *** I D M S
VERB=48 BIND Record REC--
>EMPOSITION ADDR=00040B60 Caller=CA02IDMS DMLSEQ=000010 *** I D M S
VERB=48 BIND Record REC--
>EMPLOYEE ADDR=00040B80 Caller=CA02IDMS DMLSEQ=000011 *** I D M S
VERB=48 BIND Record REC--
>DEPARTMENT ADDR=00040BF8 Caller=CA02IDMS DMLSEQ=000012 *** I D M S
CLAIM ADDR=00040C30 Caller=CA02IDMS DMLSEQ=000013 *** I D M S

```

The first page of the CA-IDMS report displays the following information:

- [A]
Displays the genlevel and version of CA IDMS/DB.
- [B]
Displays the userid used to submit the job and the job type.
- [C]
Displays the DBNODE, DBNAME, DICTNODE, and DICTNAME that were used.
- [D]
Displays the SYSIDMS parameters that were used.
- [E]
Displays the subschema name.
- [F]
Displays a trace of DML verbs for the subschema (the number of commands displayed in the trace is a user-specifiable option).

Currencies for all areas, records, and sets in the subschema at the time of the ABEND are displayed.

An example follows:

```

*****
* Currencies for subschema=EMPSS01 *
* Compiled: 2007-08-07 13.51.57 *

```

```

Current DBKEY=X'01250505' (75013:5)      For Area EMP-DEMO-REGION
Current DBKEY=X'01259801' (75160:1)      For Area ORG-DEMO-REGION

Current DBKEY=X'01259801' (75160:1)      For Record DEPARTMENT
Current DBKEY=X'01250505' (75013:5)      For Record EMPLOYEE

Current DBKEY=X'01250505' (75013:5)      For Set DEPT-EMPLOYEE
Owner DBKEY=X'01259601' (75158:1)        For Set DEPT-EMPLOYEE
Prior DBKEY=X'01251701' (75031:1)        For Set DEPT-EMPLOYEE
Next DBKEY=X'01251809' (75032:9)         For Set DEPT-EMPLOYEE

Current DBKEY=X'01250505' (75013:5)      For Set EMP-COVERAGE
Owner DBKEY=X'01250505' (75013:5)        For Set EMP-COVERAGE
Prior DBKEY=X'01256404' (75108:4)        For Set EMP-COVERAGE

```

Buffer information is displayed for each record in the subschema, as shown:

Record Name	Record Size	Bind Buff	'Prev' Buff	'Curr' Buff
-----	-----	-----	-----	-----
[A]	[B]	[C]	[D]	[E]
COVERAGE	20	00040FD8	* none *	* none *
DENTAL - CLAIM	932	00040C30	* none *	* none *
DEPARTMENT	56	00040BF8	363BC888	363BC8C0
EMPLOYEE	120	00040B80	363BC908	363BC980
EMPOSITION	32	00040B60	* none *	* none *
EXPERTISE	12	00040B50	* none *	* none *
HOSPITAL - CLAIM	300	00040A20	* none *	* none *
INSURANCE - PLAN	132	00040998	* none *	* none *
JOB	296	00040870	* none *	* none *
NON-HOSP - CLAIM	1052	00040450	* none *	* none *
OFFICE	76	00040400	* none *	* none *
SKILL	76	000403B0	* none *	* none *
STRUCTURE	12	000403A0	* none *	* none *

The buffer information consists of the following:

- **[A]**
Displays the name of the record.
- **[B]**
Displays the size of the bind buffer.
- **[C]**
Displays the address of the bind buffer.
- **[D]**
Displays previous buffer address.
- **[E]**
Displays current buffer address.

The buffer information is followed by displays of the previous, current, and bind record images for accessed records, as shown:

```

* Previous image for record DEPARTMENT      * [A]

```

- 253/326

```

3645A6D4 000000 E5C1D9E2 40404040 C5D4D7E2 E2F0F140 00000000 3641DC98 00000000 000
00000 *VARS EMPSS01 .....q.....*
3645A6F4 000020 3641DC98 00000000 04038003 01250505 01250505 3643AA70 36459008 364
1480C *...q.....*
3645A714 000040 00000000 00000000 00000000 00000000 FFFFFFFF 000124F9 0001252A 000
1252A *.....9.....*
3645A734 000060 000124FE 0001252A 0001252A 000124FA 000124FD 000124FD 04038003 FFF
FFFFF *.....*
3645A754 000080 FFFFFFFF 00000000 36459008 364149E0 00000000 00000000 00000000 000
00000 *.....\.....*
3645A774 0000A0 FFFFFFFF 0001255D 00012575 00012575 00012562 00012575 00012575 000
1255E *.....).....*
3645A794 0000C0 00012561 00012561 04038003 01259801 01259801 3643A9D8 36459008 364
14BB4 *.../.../.....q...q...z0.....*
3645A7B4 0000E0 00000000 00000000 00000000 00000000 FFFFFFFF 0001258F 000125A7 000
125A7 *.....x...x*
3645A7D4 000100 00012594 000125A7 000125A7 00012590 00012593 00012593 FFFFFFFF FFF
FFFFF *...m...x...x.....l...l.....*
3645A7F4 000120 FFFFFFFF 00000000 00000000 00000000 00000000 00000000 00000000 FFF
FFFFF *.....*
3645A814 000140 0005B0B0 FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 00000000 000
00000 *.....*
3645A834 000160 00000000 00000000 FFFFFFFF 0005B0B0 FFFFFFFF FFFFFFFF FFFFFFFF 000
00000 *.....*
3645A854 000180 00000000 00000000 00000000 00000000 00000000 FFFFFFFF 0005B0B0 FFF
FFFFF *.....*

```

Finally, CA SymDump Batch shows the following areas:

SUBSCHEMA=EMPSS01

Compiled=2007-08-07 13.51.57 [A]

Subschema Structure is Network and Unbound [B]

Area Name	Segment
EMP-DEMO-REGION /a	n
INS-DEMO-REGION	n/a [C]
ORG-DEMO-REGION /a	n

Record Name Length Procedures	Stored	Rec ID	Area Name	Data Length	Prefix
COVERAGE	VIA	400	INS-DEMO-		
REGION 20					
DENTAL-CLAIM	VIA	405	INS-DEMO-		
REGION 936					
DEPARTMENT	CALC	410	ORG-DEMO-		
REGION 56 [D]					
EMPLOYEE	CALC	415	EMP-DEMO-		
REGION 120					
EMPOSITION	VIA	420	EMP-DEMO-		
REGION 32					
EXPERTISE	VIA	425	EMP-DEMO-		

```

REGION 12          20
HOSPITAL-CLAIM      VIA 430    INS-DEMO-
REGION 300          8      CALC 435    INS-DEMO-
INSURANCE-PLAN
REGION 132          8      CALC 440    ORG-DEMO-
JOB                  IDMSCOMP Before STORE
REGION 300          24

```

IDMSCOMP Before MODIFY

IDMSDCOM After GET

```

.      .
.      .
.      .

```

```

Chain Sorted-> CALC          Next,
Prior
Owner -----
> SR1          Next=00 Prior=04

Member -----> SR6  [E]          Next=00 Prior=04

Member -----
> SR7          Next=00 Prior=04

Ckey Offset=16 Length=16 Data Type=Character

Member -----
> DEPARTMENT          Next=00 Prior=04

Ckey Offset=16 Length=4 Data Type=Numeric (Unsigned)

Member -----
> EMPLOYEE          Next=00 Prior=04

Ckey Offset=72 Length=4 Data Type=Numeric (Unsigned)

Member -----> INSURANCE-
PLAN          Next=00 Prior=04

Ckey Offset=8 Length=3 Data Type=Character

Member -----
> JOB          Next=00 Prior=04

Ckey Offset=28 Length=4 Data Type=Numeric (Unsigned)

Member -----
> OFFICE          Next=00 Prior=04

Ckey Offset=16 Length=3 Data Type=Character

Member -----
> SKILL          Next=00 Prior=04

Ckey Offset=20 Length=4 Data Type=Numeric (Unsigned)

```

```

Chain Last --> COVERAGE-CLAIMS          Next,
Prior
Owner -----
> COVERAGE          Next=12 Prior=16

Via Member --> DENTAL-
CLAIM          Next=00 Prior=04

Via Member --> HOSPITAL-
CLAIM          Next=00 Prior=04

```

Via Member --> NON-HOSP-CLAIM

Next=00 Prior=04

The preceding screen consists of the following information:

- [A]
Displays the compile date and time of the subschema.
- [B]
Displays the subschema structure and whether it is bound or unbound.
- [C]
Displays the associated segment for each area, if applicable.
- [D]
Displays detailed information about each record in the subschema, including storage mode, record ID, area name, data length, prefix length, and db procedures.
- [E]
Displays detailed information about each set in the subschema, including owner record, member records, set type, and set pointers.

Snap Report

The Snap report lets you view a snapshot of a program's data areas from various points of execution without forcing the program to abend. Each report is produced by calling the snap interface program, CAODSNAP, from any COBOL, PL/I, or Assembler program. Much of the information in this report is similar to the information provided in an abend report; however, only the most active program (the one which called CAODSNAP) is included in the report.



Note: Each call to CAODSNAP generates a separate Snap report in your central VSAM repository. If your program contains multiple snap calls (or a snap call within a loop), you may want to write the output to the CAIPRINT DD alone, to avoid *flooding* the repository.

Snap Page

The Snap report begins with some specific information pertaining to the snap.

```
*****
* SNAP OF CA02DEMO *
*****
```

```
COMPILED ON 23 APR,2004 AT 12.02.38 WITH COBOL II 1.4.0
COMPILE OPTIONS: ADV, APOST, NOAWO, NOCMR2, DATA(31), DBCS, NODECK, NODUMP,
                  DYNAM, NOFASTSRT, NOFDUMP, NOLIB, LIST, MAP, NONAME, NONUMBER,
                  OBJECT, NOOFFSET, NOOPTIMIZE, NUMPROC(PFD), RENT, RESIDENT,
                  NOSEQUENCE, SOURCE, NOSSRANGE, NOTERM, NOTEST, TRUNC(STD),
                  NOVBREF, NOWORD, XREF, ZWB
```

```
SYMBOLIC INFORMATION RETRIEVED FROM AD1DEV.MIKED.PROTSYM
MEMBER: CA02DEMO DATE: 23 APR,2004 TIME: 12.02.38 TYPE: PROTSYM
```

```
LINKED ON 23 APR 2004 AT 00:00:00
LOAD LIBRARY: MIKED.QA.LOAD
MODULE LENGTH: 00006528 (DECIMAL 25,896)
```


LINK OPTIONS: AC(0), AMODE(31), NOOVLY, NORENT, NOREUS, RMODE(24)

 * LAST STATEMENT BEFORE SNAP *

PROGRAM: CA02DEMO OFFSET: 004F92 LINE: 001413

IF PARM-MODE = 'DTE'

LINE# /LOCATION	LEVEL/FIELD NAME DEFINITION	VALUE
000157 MODE	03 PARM- *DTE*	X(3)

SNAP WAS CALLED FROM ADDRESS 0E505A80 AT OFFSET +004FA8 IN PROGRAM CA02DEMO
 SNAP COUNT IS 1

ENTRY POINT ADDRESS IS 0E500AD8 AT OFFSET +000000 IN PROGRAM CA02DEMO
 ENTRY COUNT IS 1

The contents of a Snap report depends on the language of the program that called the snap interface program CAODSNAP. The preceding screen was produced by a call from a COBOL program. For all languages, program information is provided only for the most active program, but may include multiple active procedures for PL/I.

The Snap report contains the following program information:

- **COBOL**

- Compile and link statistics
- Last statement or instruction before the snapshot
- Data division displays
- Registers
- PGT, TGT, and DSA

- **PL/I**

- Compile and link statistics
- Last statement executed in each active procedure
- Variables for each active procedure
- Last registers for each active procedure
- DSA for each active procedure
- Static and external storage dump
- Program storage dump

- **Assembler**

- Link statistics
- Program status word
- Registers, including access registers
- Addressable storage displays
- Program storage dump

If the application is connected to an IMS, DB2, or CA IDMS database at the time of the Snap call, the appropriate database section will be included in the Snap report.

CAIOPTS File Processing Report

The CAIOPTS File Processing report displays the input options read from the data set defined by the CAIOPTS DD, if present. The report displays each 80-byte record exactly as they appear in the CAIOPTS data set.

```

.....1.....2.....3.....4.....5.....6.....7.....
8
  PRTLIB CAI.
PRTLIB

  PRTREPT BOTH

  REGMAX (128,256)
  SYMDSN CAI.PROTSYM
  DUMP OFF

CAPC012I INPUT OPTION DISPLAY COMPLETE.
```

Report Summary

This report is produced during termination. The report lists alphabetically the options that were in effect at the time the report was generated.

The operating system is displayed at the top of the report as z/OS.

An example of the Report Summary appears next:

EXECUTION ON: Z/OS

EXECUTION OPTIONS IN EFFECT:

ACB	OFF
AMB	OFF
AMBL	OFF
BINDER	ON
COBONLY	OFF
DB2ACTIV	OFF
DCB	OFF
DEB	OFF
DUMP	ON
IOB	OFF
LOGROS	ON
LOGTSO	ON
LOGUNI	OFF
PLH	OFF
PRTLIB	CAI.PRTLIB

FORMATTING OPTIONS IN EFFECT:

ASMINST	ON
BINFRMT	DEC
FILES	ON
FIRST128	OFF
GRPADDR	ON
LINKAGE	ALL
LISTLINE	0
LOCALSTOR	ALL
MEMMAP	ON
MERGEDB	ON
MRGAUTO	ON
MRGBASED	ON
MRGCNTLD	ON
MRGDSECT	ON
MRGFILES	ON

PRTREPT	BOTH	MRGLINK	ON
READLL	ON	MRGLOCAL	ON
REGMAX	(128,256)	MRGPARDS	ON
RPL	OFF	MRGSTAT	ON
SAVEHEAP	OFF	MRGWORK	ON
SCRSZ80	OFF	NDVRASM	OFF
SNAP	0	NDVRDSN	
SYMDSN	CAI . PROTSYM	NDVRDSS	OFF
UCB	OFF	OCCURS	1
VSAMCAT	OFF	SAVEAREA	ON
VSAMIDX	OFF	SHOWHEAP	OFF
WTO	ON	SHOWUNMRG	TBLS
		WORKSTOR	ALL

Merged Versus Dumped Data Displays

CA SymDump Batch displays all of the storage areas for your COBOL, PL/I, or Assembler program. If symbolic information is available for the program, a variety of merging options are provided for mapping variable names and definitions onto the corresponding storage areas. Otherwise, storage areas are displayed in dump format.

- [Merged Display \(see page 259\)](#)
- [Dumped Display \(see page 262\)](#)

Merged Display

The following list describes the merging options:

- **MERGEDB ON|OFF**
Merges symbolic names onto database reports.
- **MRGAUTO ON|OFF**
Merges symbolic names onto PL/I automatic storage.
- **MRGBASED ON|OFF**
Merges symbolic names onto PL/I based variable storage.
- **MRGCNTLD ON|OFF**
Merges symbolic names onto PL/I controlled storage.
- **MRGDATA ON|OFF**
Merges symbolic names onto all data areas.
- **MRGDSECT ON|OFF**
Merges symbolic names onto Assembler DSECT storage.
- **MRGFILES ON|OFF**
Merges symbolic names onto file section for COBOL.
- **MRGLINK ON|OFF**
Merges symbolic names onto linkage section for COBOL.
- **MRGLOCAL ON|OFF**
Merges symbolic names onto local storage for COBOL.

- **MRGPARDS ON|OFF**
Merges symbolic names onto PL/I parameter storage.
- **MRGSTAT ON|OFF**
Merges symbolic names onto PL/I static storage.
- **MRGWORK ON|OFF**
Merges symbolic names onto working storage for COBOL.

The following example shows a merged data display for the WORKING-STORAGE section of a COBOL program.

*****				*****	
* WORKING-STORAGE SECTION *					

[A]	[E]				[D]
LINE#	LEVEL/FIELD NAME		VALUE		
/LOCATION			DEFINITION		
000136	77 FILLER		[G]	*THIS PROGRAM IS PROPRIETARY PROP*	
X(80)			(+000032)	*ERTY OF COMPUTER ASSOCIATES INTE*	
[B]	[C]		(+000064)	*RNATIONAL, INC. *	
000139	77 FILLER			*COPYRIGHT (C) 1986-	
1999 BY COMP*	X(80)		[H]	(+000032) *UTER ASSOCIATES INTERNATIONAL, I*	
			(+000064)	*NC. *	
000142	77 ERROR-FLAG	ZEROS			X(1)
000143	77 DATA-PICX10	ALL '&'			X(10)
000144	77 ROW-3D	1			99 BINA
RY					
000145	77 COLUMN-3D	1			99 BINARY
000146	77 RANGE-3D	1			99 BINARY
000148	01 PARAMETER-ONE			BLW=0000+0000D0 (0E77A528) [F]	
000149	03 FILLER			*PARAMETERONE*	
X(12)					
000154	01 PARM-PASSED-BY-JCL			BLW=0000+0000E0 (0E77A538)	
000155	03 PARM-MSG	*WSINIT *			X(8)
000156	03 PARM-FLAG	*WSINIT *			X(8)
000157	03 PARM-MODE	*DTE*			X(3)
000159	01 TABLE-1			BLW=0000+0000F8 (0E77A550)	
000160	05 ARRAY				
(1)		BLW=0000+0000F8 (0E77A550)			OCCURS 5
000161	10 DATA-NAME1				
(1)		ALL '&'			XXX
000162	10 DATA-NAME2(1) [I]			ALL '&'	
XXX					
	<UNMERGED DATA FOR ARRAY> [J]	X		'50505050505050505050505050505050'	
LENGTH	24				
		(+000016)	X	'5050505050505050'	
:	:				
:	:				

```
000315 01 LIST-  
IM BLW=0001+003860 (0E77ECB8) PACKED-  
DECIMAL  
000316 03 LIST-1  
(1) ? X '50505050' S9  
(6) PACKED-DECIMAL OCCURS 100  
      <UNMERGED DATA FOR LIST-  
1> X '50505050505050505050505050505050' LENGTH 396  
      LINES 000016-000368 SAME AS ABOVE  
      (+000384) X' 50505050505050505050505050'  
      473380382  
      INX  
INDEX=0001 [K]
```

A merged data display for the WORKING-STORAGE section of a COBOL program contains the following information:

- **[A]**
Displays the header describing the report columns.
- **[B]**
Displays the source line number where the data item was defined.
- **[C]**
Displays the level number and name of the data item.
- **[D]**
Displays the definition for the field. For COBOL, this includes the PICTURE and USAGE clauses and any REDEFINES or OCCURS clauses for the field.
- **[E]**
For some fields, indicates one of the following USAGE types of the data item:
 - COMP and COMP-4 items are identified as BINARY.
 - COMP-1 items are identified as FLOATING-POINT.
 - COMP-2 items are identified as LONG FLOATING-POINT.
 - COMP-3 items are identified as PACKED-DECIMAL.
 - DISPLAY-1 items are identified as DBCS.
- **[F]**
Displays the base locator and displacement of the item for group items. The address of the item is also identified, in parentheses. (The base locator prints as BL, BLL, or SBL for OS/VS COBOL and as BLF, BLW, BLL, BLK, BLV, or BLX for COBOL II and above. For COBOL II and above, the displacement may exceed 4 KB.) If the GRPADDR option is set to OFF, this information is suppressed.
- **[G]**
Displays the value of the item for elementary items. The value is displayed in a format determined by the USAGE of the data item and the validity of the data.
For example:
 - DISPLAY data is shown in display format

- PACKED-DECIMAL, BINARY, and INDEX data are shown in decimal display format (DEC) or hexadecimal format (HEX), depending on the value of the BINFRMT option.
- POINTER, PROCEDURE-POINTER, FLOATING-POINT and DBCS data are shown in hexadecimal format (for example, X'nnnnnnnn').

If the value of a data item is invalid, a question mark(?) precedes the data value and the data is displayed in hexadecimal format.

If the address of a data item is invalid, a question mark(?) precedes the data value column. The words 'INVALID ADDRESS' are displayed in the value column, followed by the address, in parentheses.

- **[H]**
Displays the offset of the continued data (from the beginning of the data item) in decimal, on each successive line. If the data is too long to fit on the current line (32 bytes for display format, 16 bytes for hexadecimal format) it is continued on the next line.
- **[I]**
Displays the first *n* occurrences of the table in merged format (where *n* is the OCCURS option value) if the value specified in the OCCURS option is less than the number of occurrences defined to the table. The rest of the table is identified as '<UNMERGED DATA FOR tablename>'. The length is displayed in the DEFINITION column and the data is displayed in hexadecimal format. The OCCURS option was set to the default value of 1 for this report. The maximum value for the OCCURS option is 16777215.
- **[J]**
Indicates the mapping status of the data in COBOL programs. If the COBOL program contains data that does not map to a data item (such as slack bytes or unmerged data items from a COPY... SUPPRESS statement), the data is identified as '<UNMERGED DATA>'. The length is displayed in the DEFINITION column and the data is displayed in hexadecimal format.
- **[K]**
Displays the index name, cell number, and current value on the line following the OCCURS data item if an INDEXED clause is specified on an OCCURS data item. If the current index value cannot be determined, 'N/A' is displayed in the value column.

When displaying a variable length table, if the size of the table has not yet been defined or cannot be determined, the maximum table size is used.

Dumped Display

An example of a dumped report follows:

* WORKING-
STORAGE SECTION *

ADDRESS[A]32FB7128[B]LENGTH 00000DAE (DECIMAL 3,502)

```
BLW=0000 +000000 F0F00000 00000000 00000000 00000000 [E]
F0F5C000 00000000 00000000 00000000[F]*00.....05{.....*
[C] [D]+000020 F0F5F000 00000000 003B0000 00000000 000A0000 00000000 00000000 0
0000000 *050.....*
+000040 001C0000 00000000 002B0000 00000000 00070000 00000000 F0000000 0
```

```

0000000 *.....0.....*
+000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0
0000000 *.....*
+000080 00010000 00000000 00000000 00000000 00000000 00010000 00000000 0
0000000 *.....*
+0000A0 00000000 00000000 00000000 00000000 00000000 FFFF0000 00000000 FFFF0000 0
0000000 *.....*
+0000C0 00050000 00000000 00000000 00000000 00000000 00000000 00000000 E5D7E4E3 4
0404040 *.....VPUT *
+0000E0 E5C7C5E3 40404040 C3C8C1D9 40404040 E5D9C5D7 D3C1C3C5 E5C4C5C6 C
9D5C540 *VGET CHAR VREPLACEVDEFINE *

```

A dumped report contains the following information:

- **[A]**
Displays the address of the dumped storage.
- **[B]**
Displays the length of the dumped storage in decimal and hexadecimal.
- **[C]**
Displays the base locator cell associated with the dumped storage (The base locator prints as BL, BLL, or SBL for OS/VS COBOL and as BLF, BLW, BLL, BLK, BLV, or BLX for COBOL II and above.)
- **[D]**
Displays the displacement of the storage from the specified base locator, in hexadecimal. (For COBOL II and above, the displacement can exceed 4 KB.)
- **[E]**
Displays the data in hexadecimal format.
- **[F]**
Displays the data in display format.

CAIPRINT Repository Viewer

The CAIPRINT Repository Viewer can be used to view CA [SymDump](#) Batch reports that are written to your central VSAM repository. Each report is automatically written to the repository defined in your run-time defaults member, CAOUDFRX, if the PRTREPT option is set to REPOS or BOTH.

The CAIPRINT Repository Viewer provides the following benefits:

- A single interface for viewing, printing, and maintaining CA SymDump Batch reports
- Easy access to multiple CAIPRINT repositories
- View-time control over the formatting options
- View-time addition of symbolic information to your reports
- Utilities for adding, viewing, printing, and maintaining symbolic information

During installation, the ISPF Primary Option menu must be updated to include a selection for the CAIPRINT Repository Viewer. If this is not completed, see [Installing](https://docops.ca.com/display/CAITSD11/Installing) (<https://docops.ca.com/display/CAITSD11/Installing>) for additional information.

Select the appropriate option from the menu to start the viewer. The Report Index panel displays.

Report Index

The Report Index panel displays all CA SymDump Batch reports that are written to the selected repository, sorted in descending order by date and time.

- [Select a Repository](#) (see page 264)
- [Report Index Fields](#) (see page 265)
- [Primary Commands](#) (see page 266)
- [Line Commands](#) (see page 268)
- [Delete a Report](#) (see page 268)
- [Lock and Unlock Reports](#) (see page 268)
- [Use the Electronic Notepad](#) (see page 269)
- [Modify Formatting Options](#) (see page 269)
- [Print a Report](#) (see page 271)
- [Select a Report for Viewing](#) (see page 273)

```

CA SymDump Batch ----- Report Index ----- Line 1 of 68
Command ==> Scroll ==> CSR
CAPI100I Repository contains 68 report(s) -----
Repository Dsname: CAI.PRTLIB
      1754 Data Records      680 Used      38.7% Full
Filter: Jobname *      Step/Tsk *      Program *      Offset *
       Comp *      UserID *      Lock *      System *
-----
Cmd  JobName  Step/Tsk  Program  Offset  Comp  Date      Time  UserId  Lvl 1
.   WANDA07D  TESTCOB2  SUBCOB   000026A S=0C7  2007/03/05  12.41  TSTUSR7
.   USRSC01A  STEP1     CARXDEMA 000013C S=0C7  2007/03/02  18.35  TSTUSR1
.   USER002R  RUN       n/a       n/a      RC=0000 2007/03/02  18.34  TSTUSR3
.   WANDA07D  TESTCOB2  SUBCOB   000026A S=0C7  2007/03/02  18.20  TSTUSR7
.   WANDA07C  TESTCOB2  SUBCOB   000026A S=0C7  2007/03/02  17.09  TSTUSR7
.   WANDA07C  TESTCOB2  SUBCOB   000026A S=0C7  2007/03/02  16.25  TSTUSR7
.   WANDA07A  S0C1     RENTSCAN 0000AFA S=0C2  2007/03/01  11.00  TSTUSR7
.   USER002A  RUN       CAOEDemo 0003D48 S=0C7  2007/03/01  10.41  TSTUSR3
.   USER002A  RUN       CAOEDemo 0003D48 S=0C7  2007/03/01  10.34  TSTUSR7
.   USER002A  RUN       CAOEDemo 0003D48 S=0C7  2007/02/28  10.06  USER01
.   WANDA07Z  S0C1     TESTSCAN 0000AFA S=0C2  2007/02/28  09.43  USER01
.   USRSC01B  RUN       CARXDEMA 000013C S=0C7  2007/02/23  14.34  USER02
.   USRSC01A  RUN       CEEPLPKA 00BAE7A U=0666 2007/02/23  14.33  USER02
.   USRSC01A  STEP1     CARXDEMA 000013C S=0C7  2007/02/23  14.28  USER02
.   USRSC01A  RUN       ABEND1    0000014 U=0001 2007/02/21  15.26  TSTUSR2

```

When the viewer is started for the first time, the repository data set name is determined using your installation defaults. For all subsequent executions, the value is retrieved from your personal user profile.

Select a Repository

You have an option of selecting a repository of your choice.

Follow these steps:

1. Use the SETINDEX command to switch to a different repository.

You are prompted for the new repository name as shown in the following screen:

```

CA SymDump Batch----- Report Index ----- Line 1 of 68
Command ==>                                         Scroll ==> CSR
-----
Repository +-----+
1754 Da |
Filter: Jo | Enter Repository Dsname: |
Co       | ==> |
-----+-----+-----+-----+-----+-----+-----+-----+-----+
Cmd  JobNa |                                     |      Lvl 1
.      WANDA +-----+-----+-----+-----+-----+-----+-----+-----+-----+
.  USRSC01A STEP1   CARXDEMA  000013C  S=0C7   2007/03/02  18.35  TSTUSR7
.  USER002R RUN    n/a       n/a       RC=0000 2007/03/02  18.34  TSTUSR3
.  WANDA07D TESTCOB2 SUBCOB   000026A  S=0C7   2007/03/02  18.20  TSTUSR7
.  WANDA07C TESTCOB2 SUBCOB   000026A  S=0C7   2007/03/02  17.09  TSTUSR7
.  WANDA07C TESTCOB2 SUBCOB   000026A  S=0C7   2007/03/02  16.25  TSTUSR7
.  WANDA07A S0C1    RENTSCAN  0000AFA  S=0C2   2007/03/01  11.00  TSTUSR7
.  USER002A RUN    CAOEDMO   0003D48  S=0C7   2007/03/01  10.41  TSTUSR3
.  USER002A RUN    CAOEDMO   0003D48  S=0C7   2007/03/01  10.34  TSTUSR7
.  USER002A RUN    CAOEDMO   0003D48  S=0C7   2007/02/28  10.06  USER01
.  WANDA07Z S0C1    TESTSCAN  0000AFA  S=0C2   2007/02/28  09.43  USER01
.  USRSC01B RUN    CARXDEMA  000013C  S=0C7   2007/02/23  14.34  USER02
.  USRSC01A RUN    CEEPLPKA  00BAE7A  U=0666 2007/02/23  14.33  USER02
.  USRSC01A STEP1   CARXDEMA  000013C  S=0C7   2007/02/23  14.28  USER02
.  USRSC01A RUN    ABEND1    0000014  U=0001 2007/02/21  15.26  TSTUSR2

```

2. Type the new data set name, fully qualified and enclosed in single quotation marks, or leave this field blank to view the default repository for the current system.

Report Index Fields

A static information area is displayed at the top of the display, immediately below the command line and separated by dashed lines.

The following information is displayed in this area:

- **Repository Dsname**

Displays the data set name of the currently selected repository.

- **Data Records**

Displays the number of data records available in the selected repository.

- **Used**

Displays the number of used data records in the selected repository.

- **Full**

Displays the percentage of used data records.

- **Filters**

Lists eight updatable fields (Jobname, Step/Tsk, Program, Offst, Comp, UserID, Lock, System) that can be used to filter the list of reports that can be selected.

The final static line in this display is the highlighted header line, containing all of the column headers for the report list.

The data portion of the display is horizontally and vertically scrollable and can be sorted in ascending or descending order on any column header.

For each report in the list, the following information is displayed:

- **Job**
Displays the name of the job that produced the report.
- **Step/Tsk**
Displays the name of the step that produced the report or the IMS DC task ID if the report was generated in an IMS DC environment under LE.
- **Program**
Displays the name of the abending program that caused the report to be produced.
- **Offset**
Displays the offset in the abending program where the abend occurred.
- **Comp**
Displays the completion code from the step that produced the report.
- **Date**
Displays the date on which the report was produced.
- **Time**
Displays the time when the report was produced.
- **Userid**
Displays the name of user who submitted the job that produced the report.
- **Lock**
Displays the name of the user that owns a lock on the report to prevent deletion.
- **System**
Displays the name of the system on which the report was created.
- **Asid**
Displays the address space ID of the job which created the report.
- **#Recs**
Displays the number of physical repository records occupied by the report.
- **S I P**
Indicates whether the report contains Snaps, Intercepts, or Parmchk.

Primary Commands

The following primary commands are available from the Report Index panel:

- **BOTTOM**
Scrolls the display to the last line.

- **DOWN**
Scrolls the display forward.
- **FIND**
Locates text in the current display.
- **FM**
Invokes CA File Master Plus to view a data set.
- **HELP**
Requests help for a command, message, or topic.
- **KEEP**
Adds data from the current display to the Keep Window.
- **LEFT**
Scrolls the display to the left.
- **OPTIONS**
Displays the current installation options in effect.
- **PRINT**
Prints all or part of the current display.
- **PROFILE**
Views and updates display preferences.
- **REFRESH**
Refreshes the current display.
- **RFIND**
Repeats a previous FIND command.
- **RIGHT**
Scrolls the display to the right.
- **SET**
Selects a new repository.
- **SORT**
Sorts the current display on any column heading.
- **SYM**
Displays the Symbolic Utilities menu.
- **TOP**
Scrolls the display to the first line.
- **UP**
Scrolls the display backward.

A description of each primary command can be found later in this section.
For viewing the online help for any primary command, type HELP command-name.

Delete a Report

Follow these steps:

- You are prompted for confirmation to prevent accidental deletion as shown in the following screen:

```

CA SymDump Batch ----- Report Index ----- Line 1 of 68
Command ==> Scroll ==> CSR
-----
Repository +-----+
  1754 Dat | Press ENTER to confirm the delete request for | *
Filter: Job | the selected report: | *
  Com      | | |
-----+-----+
Cmd  JobNam | Program: CARXDEMA Offset: 000013C Comp: S=0C7 | ----- Lvl 1
      | Created: 2007/02/23 at 14.28 | e   UserId
.    WANDA0  | | | | 41  TSTUSR7
.    USRSC0  | Press END to cancel the delete request. | 35  TSTUSR1
.    USER00  | | | | 34  TSTUSR3
      WANDA0  | Display this message? (Y/N) ==> Y | 20  TSTUSR7
      WANDA0  +-----+ | 09  TSTUSR7
.    WANDA07C TESTCOB2 SUBCOB 000026A S=0C7 2007/03/02 | 16.25 TSTUSR7
.    WANDA07A S0C1  RENTSCAN 0000AFA S=0C2 2007/03/01 | 11.00 TSTUSR7
.    USER002A RUN  CAOEDEMO 0003D48 S=0C7 2007/03/01 | 10.41 TSTUSR3
.    USER002A RUN  CAOEDEMO 0003D48 S=0C7 2007/03/01 | 10.34 TSTUSR7
.    USER002A RUN  CAOEDEMO 0003D48 S=0C7 2007/02/28 | 10.06 USER01
.    WANDA07Z S0C1  TESTSCAN 0000AFA S=0C2 2007/02/28 | 09.43 USER01
.    USRSC01B RUN  CARXDEMA 000013C S=0C7 2007/02/23 | 14.34 USER02
.    USRSC01A RUN  CEEPLPKA 00BAE7A U=0666 2007/02/23 | 14.33 USER02
d    USRSC01A STEP1 CARXDEMA 000013C S=0C7 2007/02/23 | 14.28 USER02
.    USRSC01A RUN  ABEND1 0000014 U=0001 2007/02/21 | 15.26 TSTUSR2

```

- You can suppress confirmation for the remainder of the session by entering **N** in the Display this message? field located at the bottom of the confirmation window.

Use PROFILE command to suppress all delete confirmations for your ID.

Lock and Unlock Reports

You can lock a report to ensure that it is not edited or deleted by other users.

Follow these steps:

- Your userid appears in the Lock column, indicating that you have requested a lock for the report. While a report is locked, it cannot be deleted from the repository.

2. Type the U line command to unlock a report that was previously locked.

Use the Electronic Notepad

You can use the Electronic Notepad as a scratch pad while debugging a problem.

To use the electronic notepad, type the N line command to open the Electronic Notepad for a report.

Your notes are stored in your user profile, are unique to the selected report, and can only be viewed from your own userid.



Note: For more information about a complete description of the Electronic Notepad, see [Advanced Techniques \(see page 277\)](#).

Modify Formatting Options

When a report is created, your installation defaults and CAIOPTS overrides are used to select formatting options for the report. Using the Repository Viewer, you can change many of these formatting options dynamically to control the format and content of your reports.

Follow these steps:

1. Type the O line command to display the Report Options panel.

The following screen is displayed:

```

CA SymDump Batch ----- Report Options ----- Line 1 of 15
Command ==>                                         Scroll ==> CSR
-----
JobName  USER002A  Step/Tsk  RUN      UserId  USER002  Date  2007/01/25
Program  CAOEDemo    Offset  0003D48  Comp    S=0C7    Time  12.40
-----
Display Options                                Merging Options
ASMINST  ==> ON                                BINFRMT  ==> DEC
FILES    ==> ON                                GRPADDR  ==> ON
FIRST128 ==> OFF                                MERGEDB   ==> ON
LINECNT  ==> 60                                MRGAUTO   ==> ON
LINKAGE  ==> ON                                MRGBASED  ==> ON
LISTLINE ==> 0                                  MRGCNTLD  ==> ON
LOCALSTOR ==> ON                                MRGDSECT  ==> ON
MEMMAP   ==> ON                                MRGFILES  ==> ON
RPTSZ80  ==> OFF                                MRGLINK   ==> ON
SAVEAREA ==> ON                                MRGLOCAL  ==> ON
SHOWHEAP ==> OFF                                MRGPARDS  ==> ON
WORKSTOR ==> ON                                MRGSTAT   ==> ON
                                                MRGWORK   ==> ON
                                                OCCURS    ==> 1
                                                SHOWUNMRG ==> TBL5
----- Lvl 2

```

2. Modify the following formatting display and merging options as required:

- **Display Options:**

- **ASMINST ON|OFF**

Displays the abending Assembler instruction.

- **FILES ON|OFF**
Displays the Open Files report.
- **FIRST128 ON|OFF**
Displays only the first 128 bytes of each record.
- **LINECNT *nnn***
Displays the number of lines per formatted page.
- **LINKAGE ON|OFF**
Displays the linkage section for COBOL programs.
- **LISTLINE *nn***
Displays the number of additional listing lines to be merged into a report before and after the source statement at abend or snap.
- **LOCALSTOR ON|OFF**
Displays local storage for COBOL programs.
- **MEMMAP ON|OFF**
Displays the TGT, DSA, and PGT memory maps.
- **RPTSZ80 ON|OFF**
Formats when possible using 80-byte output.
- **SAVEAREA ON|OFF**
Displays the Save Area Trace report.
- **SHOWHEAP ON|OFF**
Displays LE or PL/I heap storage.



Note: SHOWHEAP ON requires SAVEHEAP ON at execution time for LE enabled COBOL or Assembler programs.

- **WORKSTOR ON|OFF**
Displays the working storage for COBOL programs.
- **Merging Options:**
- **BINFRMT DEC|HEX**
Displays binary values as decimal or hexadecimal.
- **GRPADDR ON|OFF**
Displays the group address for 01-items.
- **MERGEDB ON|OFF**
Merges symbolic names onto Database reports.
- **MRGAUTO ON|OFF**
Merges symbolic names onto PL/I automatic storage.

- **MRGBASED ON|OFF**
Merges symbolic names onto PL/I based variable storage.
- **MRGCNTLD ON|OFF**
Merges symbolic names onto PL/I controlled storage.
- **MRGDSECT ON|OFF**
Merges symbolic names onto Assembler DSECT storage.
- **MRGFILES ON|OFF**
Merges symbolic names onto file section for COBOL.
- **MRGLINK ON|OFF**
Merges symbolic names onto linkage section for COBOL.
- **MRGLOCAL ON|OFF**
Merges symbolic names onto local storage for COBOL.
- **MRGPARDS ON|OFF**
Merges symbolic names onto PL/I parameter storage.
- **MRGSTAT ON|OFF**
Merges symbolic names onto PL/I static storage.
- **MRGWORK ON|OFF**
Merges symbolic names onto working storage for COBOL.
- **OCCURS *nnn*|MAX**
Displays the number of table elements to display for each array.
- **SHOWUNMRG ALL|NONE|TBLS**
Displays storage for unmerged data.

3. Modify the formatting options as desired, press END.
Your changes are saved to the repository. Press CANCEL to return to the Report Index panel without saving.

Print a Report

You have an option of printing a report to a printer, file, or other destination.

Follow these steps:

1. Type the P line command to send a report to a printer, file, or other destination.
The Print Options pop-up window displays:

```

CA SymDump Batch ----- Report Index ----- Line 1 of 68
Command ==>                                         Scroll ==> CSR
-----
Repo +-----+
  1 | |
Filt | Print Options:                * Press ENTER to PRINT * |
---- | |
Cmd  | Printer    ==> SYSTEM1A.USER01 | Lvl 1
     | or        |                    | rId

```

```

P | Dsname      ==> 'USER01.RPTFILE' | USR7
. | Member      ==> NEWMEM      Disposition ==> NEW | USR1
. |             |             |             | USR3
. | Copies      ==> 001         | USR7
. | Class       ==> X           | USR7
. | PageDef     ==>             | USR7
. | FormDef     ==>             | USR7
. | Chars       ==>             | USR3
. |             |             |             | USR7
. | Format Y/N  ==> Y           | R01
. |             |             |             | R01
. |             |             |             | R02
. |             |             |             | R02
. |             |             |             | R02
. |             |             |             | R02
. |             |             |             | USR2
+-----+

```

2. Complete the following fields and press Enter.

▪ **Printer**

Lets you define the destination to which the printed output is sent. You can specify a printer name, node.userid, or leave the field blank to spool the printed output to your TSO userid.

▪ **Dsname**

Lets you define the name of a data set to which the printed output is written. If this field is non-blank, it overrides the Printer field. When entering a data set name, it must be fully qualified and enclosed in single quotation marks. Otherwise, your ZPREFIX value is used as the leading qualifier.

▪ **Member**

Lets you define the member name if you are printing the report to a partitioned data set.

▪ **Disposition**

Lets you specify the disposition of the data set to which the output should be printed. This field is relevant only if printing to a data set. If the value is NEW or MOD, a second window pops up to request allocation parameters.

Values: SHR, OLD, NEW, and MOD



Note: For a description of the fields on the window, see the PRINT command.

▪ **Copies**

Lets you define the number of copies to be printed.

▪ **Class**

Lets you define the sysout class to use to allocate the print data set.

▪ **PageDef**

Lets you define the name of the library member that PSF uses to define the page layout for printing on a 3800 Printing Subsystem Model 3.

▪ **FormDef**

Lets you define the name of the library member that PSF uses to define the form layout for printing on a 3800 Printing Subsystem Model 3.

- **Chars**

Lets you define the name of the character arrangement tables for printing on a 3800 Printing Subsystem Model 3.

- **Format**

Lets you specify whether to print reports for viewing at your installation.

Default: Y

Values:

- **Y**

Prints reports for viewing at your installation.

- **N**

Does not print reports for viewing. Set this field to N only when instructed to do so by CA Technical Support for diagnostic purposes.

3. Press CANCEL to return to the Report Index panel without printing.

Usage Notes:

- When one or more fields are changed before pressing Enter, the fields are validated and redisplayed along with the message Press Enter to Print. To continue, press Enter again.
- Reports are printed using the formatting options defined in the repository. Use the O line command prior to printing to confirm these options.

Select a Report for Viewing

To select a report for viewing, type the S line command.

When a report is selected for viewing, the report data is formatted using your customized formatting options and the available symbolic information. The report is then loaded into memory, a table of contents called the Report Tree is built, and the Report Tree panel displays.

Report Tree

The Report Tree panel displays all of the key report sections within your report, sorted in the order in which they appear. It provides a shortcut to any section within the selected report.

- [Report Tree Fields \(see page 274\)](#)
- [Primary Commands \(see page 275\)](#)
- [Line Commands \(see page 275\)](#)
- [Expand and Collapse the Tree View \(see page 276\)](#)

```
CA SymDump Batch ----- Report Tree ----- Line 1 of 10
Command ==>                               Scroll ==> CSR
CAPI052I Profile restored as of 11:31:59 on 2007/01/26 -----
JobName  USER002A  Step/Tsk  RUN      UserId  USER002      Date  2007/01/25
Program  CAOEDMO   Offset    0003D48  Comp    S=0C7        Time  12.40
----- Lvl 2
Cmd  LineNo.  Description
.    1 - Job=USER002A
```

```

.      1  | -Input Options Display
.      6 - | -Abend S=0C7 (CA0EDEM0)
.     20  | | -Abending Statement
.     34  | | -Abending Instruction
.     51  | | -Module Call Sequence
.     63 + | | -Program CA0EDEM0
.    811 + | | -Open Files
.    819  | | -Save Area Trace
.    843  | | -Execution Summary
-----

```

To instantly position your display to any section in the report, Type **S** in the Cmd column next to that section and press Enter. To position your display to the top of the report, type **S** next to the first entry in the Report Tree.

After your display is positioned within the report, you can reposition your display using the standard positioning commands (UP, DOWN, FIND, RFIND, LOCATE) or press END to return to the tree and select a different shortcut.

Report Tree Fields

A static information area is displayed at the top of the panel, immediately below the command line and separated by dashed lines.

The following information is displayed in the static information area:

- **Job Name**
Displays the name of the job for which the report was created.
- **Step/Tsk**
Displays the name of the step that produced the report or the IMS DC task ID if the report was generated in an IMS DC environment under LE.
- **Userid**
Displays the name of the user owning the job for which the report was created.
- **Date**
Displays the date on which the report was created.
- **Program**
Displays the name of the program in control at the time of the error.
- **Offset**
Displays the offset within the program at the time of the error.
- **Comp**
Displays the completion code (S=xxx, U=dddd, or RC=dddd).
- **Time**
Displays the time at which the report was captured.

The final static line in this display is the highlighted header line, containing all of the column headers for the Report Tree panel.

The data portion of the display is scrollable but cannot be sorted.

For each entry in the Report Tree, the following information is displayed:

- **LineNo**
Displays the line number within the report where the report section corresponding to the tree entry begins.
- **Description**
Displays a brief description of the report section, generally matching the header found at the top of the section in the report.

Primary Commands

The following primary commands are available from the Report Tree panel:

Command	Description
BOTTOM	Scrolls the display to the last line.
DOWN	Scrolls the display forward.
FIND	Locates text in the current display.
FM	Invokes CA File Master Plus to view a data set.
HELP	Requests help for a command, message or topic.
KEEP	Adds data from the current display to the Keep Window.
LEFT	Scrolls the display to the left.
OPTIONS	Displays the current installation options in effect.
PRINT	Prints all of part of the current display.
PROFILE	Views and updates display preferences.
RFIND	Repeats a previous FIND command.
RIGHT	Scrolls the display to the right.
SYM	Displays the Symbolic Utilities menu.
TOP	Scrolls the display to the first line.
UP	Scrolls the display backward.
VIEW	Displays symbolic listing for a program.

A description of each primary command can be found later in this section .

For viewing the online help for any primary command, type **HELP command-name**.

Line Commands

The following line commands are available from the Report Tree panel:

Command	Description
+	Expands an entry in the tree to view its subordinate sections.
-	Collapses an entry in the tree to hide its subordinate sections.
!	

Command Description	
	Explodes an entry in the tree by first expanding the entry and then expanding each of its subordinates, recursively, until all of the sections are visible.
/	Toggles an entry between the expanded and collapsed views.
S	Selects a report section for viewing.

Expand and Collapse the Tree View

Some report sections are logical subsets of other larger sections, and this relationship is represented in the tree using indentation and an increasing number of vertical bars.

In the following example, an Abend report for program SBDEMOC, which begins on line eight, has six subordinate report sections.

```
.      8-      | -Abend S=0C7 (SBDEMOC)
.      22 |  | -Abending Statement
.      36 |  | -Abending Instruction
.      53 |  | -Module Call Sequence
.      65+    |  | -Program SBDEMOC
.      802+   |  | -Open Files
.      810 |  | -Save Area Trace
```

The Abend report for program SBDEMOC is displayed with only one vertical bar. Each of the six subordinate sections are displayed with two vertical bars, demonstrating that they are each logically part of the Abend report.

A section in the report tree is expanded when its subordinate sections are visible in the display and collapsed when its subordinate sections are not visible. An expanded section is identified in the display by the - symbol immediately to the right of the line number, while a collapsed section is identified by a + symbol. When neither symbol appear, the report section has no subordinate sections.

This example shows the same report tree, except that the entry for program SBDEMOC is expanded to show its subordinate sections. This view now contains a direct *shortcut* to all of this program's data sections and other key reports.

```
.      8-      | -Abend S=0C7 (SBDEMOC)
.      22 |  | -Abending Statement
.      36 |  | -Abending Instruction
.      53 |  | -Module Call Sequence
.      65-    |  | -Program SBDEMOC
.      84+    |  | -File Section
.      110 |  | -Working-Storage Section
.      575 |  | -External Working-Storage Section
.      584 |  | -Linkage Section
.      600 |  | -Registers at Abend
.      637+   |  | -Memory Maps
.      802+   |  | -Open Files
.      810 |  | -Save Area Trace
```

By expanding and collapsing the Report Tree view, you can easily zoom in to easily locate any areas of interest within your report.

View a Report

When you use the S line command from the Report Tree panel, your display is positioned directly to the selected section of the report as shown in the following screen:

```

CA SymDump Batch ----- Report ----- Line 8 of 915
Command ==> Scroll ==> CSR
-----
JobName  SBDEMO C      Step/Tsk  RUN      UserId  MIKED      Date  2004/05/17
Program  SBDEMO C      Offset   0003B94   Comp    S=0C7      Time  15.33
-----
.....1.....2.....3.....4.....5.....6.....7..... Lvl 3
*****
*          * S-0C7 *
*  A B E N D *-----*
*          * SBDEMO C *
*****
*****
*      DESCRIPTION:  S0C7 - A NUMERIC FIELD CONTAINED NON-NUMERIC DATA.
*
*  PROBABLE CAUSES:  1. NUMERIC DATA WAS NOT INITIALIZED.
*                    2. A SUBSCRIPT OR INDEX CONTAINED AN INVALID VALUE.
*                    3. A COMP-3 FIELD HAD AN INVALID SIGN.
*                    4. A GROUP MOVE OVERLAYED A NUMERIC FIELD WITH NON-NUMERI
*****
*****
*  ABENDING STATEMENT *
*****

```

Report View Fields

The static information area displayed at the top of the Report Tree panel continues to display while viewing the report.

The final static line in this display is the highlighted ruler, which scrolls left and right automatically as you scroll the data.

The data portion of the display is both vertically and horizontally scrollable when the report length or width cannot be fully accommodated by your screen dimensions.

Advanced Techniques

Several easy-to-master techniques involving simple primary commands can make your report viewing sessions more productive. We recommend that you set your PF keys to some of these primary commands to facilitate their use:

- The Keep Window and the Electronic Notepad help you with your diagnostic procedure. Use the KEEP command to copy data from any display into your Keep Window.
- The NOTES command lets you maintain notes for each report as you view.
- The TAG and LOCATE commands let you quickly maneuver through different sections of a report.

This section provides a detailed description of each of these advanced techniques.

- [Set PF Keys \(see page 278\)](#)
- [Use the Keep Window \(see page 279\)](#)
- [Use the Electronic Notepad \(see page 280\)](#)
- [Use TAG and LOCATE \(see page 282\)](#)

Set PF Keys

Using PF keys to execute some of the more common commands helps save time and reduce keystroke errors when entering commands.

Follow these steps:

1. Type **KEYS** on any Repository Viewer command line and press Enter.
The PF Keys panel displays, which lets you change your PF key settings for the product.



Note: If your Repository Viewer was invoked using the NEWAPPL keyword, it executes under its own unique application ID and therefore has its own unique set of PF key settings. Therefore, setting your PF keys for the viewer will not affect your settings for other ISPF applications.

The following screen shows the PF Keys panel:

```

                                PF Key Definitions and Labels - Primary Keys
Command ===>
Number of PF Keys . . . 24
Enter "/" to select . . . (Enable EURO sign)
Terminal type . . 3278
More: +

PF1 . . . HELP
PF2 . . . SPLIT
PF3 . . . END
PF4 . . . DUMP
PF5 . . . RFIND
PF6 . . . MAP
PF7 . . . UP
PF8 . . . DOWN
PF9 . . . SWAP
PF10 . . LEFT
PF11 . . RIGHT
PF12 . . RETRIEVE

PF1 Label . .      PF2 Label . .      PF3 Label . .
PF4 Label . .      PF5 Label . .      PF6 Label . .
PF7 Label . .      PF8 Label . .      PF9 Label . .
PF10 Label . .     PF11 Label . .     PF12 Label . .

```

2. Use the PF Keys panel to assign any PF key to any primary command.

It is recommended to only reassign those keys whose functions are not used by the viewer. For example, some of your PF keys may already be assigned to some of these functions used by the viewer:

- DOWN

- END
- HELP
- LEFT
- RFIND
- RETRIEVE
- RIGHT
- SPLIT
- SWAP

However, you can replace the following keys that are not used by the viewer:

- RCHANGE
- RETURN

We strongly recommend that you set one of your available PF keys to KEEP to facilitate the use of this cursor-sensitive command.

Use the Keep Window

The Keep Window is a dynamic area located just above the highlighted header line on any display. The size of the window is dynamic depending on the number of data lines added. When the window is empty, its borders are not displayed. You can copy any data line from any viewer display into this area, causing that data line to remain in view even after you exit the display.

Examples of data you might add to your Keep Window include:

- Text from a HELP command
- Source statements from the SYM command
- Key data values from anywhere in your report
- Compile and link information for a program

To add a data line to your Keep Window, first type **KEEP** on the command line, then place your cursor on the desired data line and press Enter. Alternatively, if you have set a PF key for KEEP, place your cursor on any data line and press that key. The data line is added automatically to your Keep Window.

The following example shows compile information for program SBDEMO in the Keep Window:

```
CA SymDump Batch ----- Report ----- Line 22 of 898
Command ==> Scroll ==> CSR
-----
JobName  SBDEMO      Step/Tsk  RUN      UserId  USER02    Date  2007/06/05
Program  SBDEMO      Offset   0003B94  Comp    S=0C7     Time  12.05
```

```

----- (Keep) -----
COMPILED ON 21 MAY,2007 AT 16.30.20 WITH COBOL FOR Z/OS 3.4.1
COMPILE OPTIONS: ADV, APOST, NOAWO, DATA(31), DBCS, NODECK, NODUMP, DYNAM,
                  NOFASTSRT, LIB, LIST, MAP, NONAME, NONUMBER, OBJECT, NOOFFSET,
                  OPTIMIZE, NUMPROC(PFD), RENT, SEQUENCE, SOURCE, NOSSRANGE,
                  NOTERM, NOTEST, TRUNC(STD), NOVBREF, NOWORD, XREF, ZWB
----- Lvl 3
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*****
* ABENDING STATEMENT *
*****

PROGRAM: SBDEMO0C   OFFSET: 003B8E   LINE: 001284

001284          MOVE MASK1  TO  BINARY-1

LINE#   LEVEL/FIELD NAME                               VALUE/LOCATION
-----
000329   03 MASK1                                     ALL  '*'

```

Each user has a separate Keep Window for each report they view. The contents of each is saved in the user profile when a report viewing session is ended and is restored to the same state the next time that same dump is selected for viewing.



Note: You can also use the KEEP command from the Report Index panel, but that window is maintained separately, is not preserved across sessions, and cannot be viewed while a report is loaded.

To remove a data line from the Keep Window, first type **KEEP** on the command line, place your cursor on the line being removed, and press Enter. If you have a PF key set for KEEP, place your cursor on the line being removed and press that key.

To prevent your Keep Window from being displayed without removing its contents, type **KEEP OFF** on the command line and press Enter. Type **KEEP ON** to restore the display at any time. The status of your window display (ON or OFF) is also saved in the user profile for each report you view.



Note: When the Keep Window status is OFF, the data in your Keep Window is not lost; it is simply not displayed until the window display is reactivated. Data cannot be added or removed from the Keep Window while its status is OFF.

On some terminals, adding several lines to your Keep Window can dramatically reduce the number of visible lines of report data. If this is a problem, you can use the PROFILE command to suppress the information area at the top of the display.

Use the Electronic Notepad

The Electronic Notepad lets users maintain notes for each report they view. The notepad is free-format and dynamically sized, which lets you enter any of your thoughts, ideas, or comments while viewing the report. You can also copy and paste report information into the notepad for easy retrieval.

Notepad records are 126 bytes wide to fill a 132-column screen format and are horizontally scrollable for 80-column displays. You can type any kind of data in the notepad because there is no syntax checking.

To open a notepad from the Report Index panel, type **N** in the Cmd column next to the report entry and press Enter.

To open a notepad for an open report while viewing, type **NOTES** on the command line and press Enter.

The first time you open the notepad for a report, the following message displays:

```

CA SymDump Batch ----- Report Index ----- Line 1 of 68
Command ==> Scroll ==> CSR
-----
Repositor +-----+
1754 D |
Filter: J | No notes were found for the report you selected.
C |
-----+-----+
Cmd JobN | Press ENTER to start a new notepad for this report. | ---- Lvl 1
N WOND | Program: SUBCOB Offset: 000026A Comp: S=0C7 | UserId
. DAVS | | TSTUSR7
. RYAR | Press END to cancel the notes request. | TSTUSR1
. WOND | | TSTUSR3
. WOND | | TSTUSR7
. WOND | | TSTUSR7
. WANDA07C TESTCOB2 SUBCOB 000026A S=0C7 2007/03/02 16.25 TSTUSR7
. WANDA07A S0C1 RENTSCAN 0000AFA S=0C2 2007/03/01 11.00 TSTUSR7
. USER002A RUN CAOEDMO 0003D48 S=0C7 2007/03/01 10.41 TSTUSR3
. USER002A RUN CAOEDMO 0003D48 S=0C7 2007/03/01 10.34 TSTUSR7
. USER002A RUN CAOEDMO 0003D48 S=0C7 2007/02/28 10.06 USER01
. WANDA07Z S0C1 TESTSCAN 0000AFA S=0C2 2007/02/28 09.43 USER01
. USRSC01B RUN CARXDEMA 000013C S=0C7 2007/02/23 14.34 USER02
. USRSC01A RUN CEEPLPKA 00BAE7A U=0666 2007/02/23 14.33 USER02
. USRSC01A STEP1 CARXDEMA 000013C S=0C7 2007/02/23 14.28 USER02
. USRSC01A RUN ABEND1 0000014 U=0001 2007/02/21 15.26 TSTUSR2

```

Press Enter to create a new notepad for the report, or press END to cancel the request.

A new notepad, like the one shown in the following screen, initially contains ten blank records:

```

CA SymDump Batch ----- Notes ----- Line 1 of 10
Command ==> Scroll ==> CSR
-----
Program: SUBCOB Offset: 000026A Comp: S=0C7
Last Update: None
-----+-----+
Cmd .....1.....2.....3.....4.....5.....6.....7.....
.
.
.
.
.
.
.
.
.
.

```

The information area located at the top of the display identifies the report for which the notepad was created.

The final static line in this display is the highlighted ruler that scrolls left and right automatically as you scroll your notes.

The data portion of the display is both vertically and horizontally scrollable when the length or width of the notepad cannot be fully accommodated by your screen dimensions.

Type the I line command to insert a new blank record following any existing record in the notepad.
Type the D line command to delete an existing record from the notepad.

When you have completed your updates, press END to save your updates and return to the display from which the notepad was entered. Use CANCEL to exit without saving your changes.

Use TAG and LOCATE

For large reports, it may be necessary to create placeholders for sections of the report that do not appear in the Report Tree panel.

Use the TAG primary command to define a tag or label for any location in your report. You are prompted for a tag name, as shown in the following screen:

```

CA SymDump Batch ----- Report ----- Line 115 of 915
Command ==>                               Scroll ==> CSR
-----
JobName  SBDEMO0C +-----+ ate 2004/05/17
Program  SBDEMO0C |         | ime 15.33
-----+-----+ |         | ----- Lvl 3
...+...1...+... | Enter Tag Name: | ...+...7...+...
000062  77 FILLER | ==>          | PROPRIETARY PROP*
+-----+-----+ |         | R ASSOCIATES INTE*
                                (+000064) *RNATIONAL, INC. *
000065  77 FILLER                                * COPYRIGHT (C) 1986-1999 BY COMP*
                                (+000032) *UTER ASSOCIATES INTERNATIONAL, I*
                                (+000064) *NC. *
000068  77 ERROR-FLAG                                ZEROS
000069  77 DATA-PICX10                                LOW-VALUES
000070  77 ROW-3D                                1
000071  77 COLUMN-3D                                1
000072  77 RANGE-3D                                1
000074  01 PARAMETER-ONE                                BLW=0000+0000D0 (121B1188)
000075  03 FILLER                                *PARAMETERONE*
000080  01 PARM-PASSED-BY-JCL                                BLW=0000+0000E0 (121B1198)
000081  03 PARM-MSG                                *NONE *
000082  03 PARM-FLAG                                *NONE *
000083  03 PARM-MODE                                *DTE*

```

Type a name for your tag, from 1 to 32 characters and press Enter to create your tag. Leading periods are ignored.

Alternatively, you can create a tag using the *.label* method by entering your tag name on the command line with a leading period. For example, to create a tag named *EF* for a variable named ERROR-FLAG, position your display to that variable and type *.EF* on the command line. Press Enter to create the tag.



Note: If you create a tag with a duplicate name, the new tag replaces the previously existing tag.

After you create your tag, you can position your display immediately to that line using the LOCATE command or L as an abbreviation. In the example above, type **L EF** to locate the variable ERROR-FLAG. If you enter a LOCATE command without any tag name, you are prompted for a tag name as shown in the following screen:

```
CA SymDump Batch ----- Report ----- Line 22 of 915
Command ==> Scroll ==> CSR
-----
JobName  SBDEMO0C +-----+ ate 2004/05/17
Program  SBDEMO0C |         | ime 15.33
-----+-----+ |         | ----- Lvl 3
          | Enter Tag Name: |         | ...+...7....+...
          | ==>             |         |
*****+-----+
* ABENDING STATEM +-----+
*****
```

PROGRAM: SBDEMO0C OFFSET: 003B8E LINE: 00128

MOVE MASK1 TO BINARY-1

LINE#	LEVEL/FIELD NAME	VALUE/LOCATION
000329	03 MASK1	ALL '*'
000178	03 BINARY-1	+1

ABENDING INSTRUCTION

4F30 D4B8 CONVERT

Tags are not stored in the user profile, so all tags are cleared when you exit a report viewing session.

Symbolic Utilities

- [List the Contents of a Symbolic File \(see page 284\)](#)
- [Delete a Member from a Symbolic File \(see page 285\)](#)
- [Print a Program Listing from a Symbolic File \(see page 286\)](#)
- [View a Program Listing from a Symbolic File \(see page 288\)](#)
- [Add a Listing to a PROTSYM File \(see page 289\)](#)
- [List Globally Defined Symbolic Files \(see page 291\)](#)
- [List Supplemental Symbolic Files \(see page 291\)](#)
- [View Dynamic Symbolic Support Options \(see page 292\)](#)
- [Override Dynamic Symbolic Support Options \(see page 293\)](#)

Use the SYM primary command from anywhere in the Repository Viewer to display the Symbolic Utilities menu as shown in the following screen:

```
CA SymDump Batch----- Report Index ----- Line 1 of 33
Command ==> Scroll ==> CSR
-----
Rep +-----+
    | Select a Symbolic Utility ==> |
---+-----+
Cmd | 1. List the contents of a symbolic file | vl 1
    | 2. Add listings to a PROTSYM file   | Id
    | 3. List globally defined symbolic files | C01
    | 4. Add or Remove supplemental symbolic files | C01
    | 5. Dynamic Symbolic Support for CA Endeavor SCM | C01
    | | | D01
    | | | 002
```

```

. | Symbolic File Dsname: | 002
. | ==> 'CAI.PROTSYM' | 002
. +-----+ 002
. USER002A RUN A02DEMO 00041B6 S=0C7 2004/03/05 14.47 USER002
. USER002S RUN PLITEST1 0001002 S=0C7 2004/03/04 16.29 USER002
. USER002S RUN PLITEST1 0001002 S=0C7 2004/03/04 16.18 USER002
. USER002P RUN PLITEST1 0001E22 S=0C7 2004/03/04 14.36 USER002
. USER002P RUN PLITEST1 0001C46 S=0C7 2004/03/04 14.35 USER002
. USRSC01A RUN COMP2 00001E8 S=0C7 2004/02/25 17.46 USRSC01
. USRSC01A RUN COMP1 00003FE S=0C7 2004/02/24 18.06 USRSC01
. USRSC01A RUN COMP1 00003FE S=0C7 2004/02/24 17.46 USRSC01
. USRSC01A RUN COMP1 00003FE S=0C7 2004/02/24 17.37 USRSC01
. USRSC01A RUN COMP2 00001E8 S=0C7 2004/02/24 11.51 USRSC01

```

The symbolic utilities can be used to:

- List the contents of your symbolic files
- Delete a member from a symbolic file
- Print a program listing from a symbolic file
- View a listing from a symbolic file
- Add a listing to your PROTSYM file
- List your globally defined symbolic files
- Add or remove your supplemental symbolic files
- Add or change DSS options for the Viewer ISPF session.

Select one of the valid options from the menu to continue.

List the Contents of a Symbolic File

Select SYM option 1 to list the contents of a symbolic file. For this option, you must also type the data set name for the PROTSYM or CSL file that you want to view. To enter a fully-qualified data set name, enclose the name in single quotation marks. Otherwise, your ZPREFIX value is appended to the name as the high-level qualifier.



Note: Every time you access a symbolic file using one of the online utilities, the data set name is stored in your profile.

After you type a symbolic file name, press Enter to list the contents of the file.

```

CA SymDump Batch----- PROTSYM Directory ----- Line 1 of 42
Command ==> Scroll ==> CSR
-----
PROTSYM Dsname: CAI.PROTSYM
----- Lvl 2
Cmd  Program   Date       Time       Size  Language  Attributes
.   APF1       2003/07/16  18.44.00   11    HLASM
.   APF2       2003/07/16  18.34.00   10    HLASM
.   ASMTTEST   2003/08/25  10.30.00   11    HLASM
.   ATTNTEST   2003/07/03  11.28.00   12    HLASM

```

.	BIGSEQ	2003/01/16	10.50.30	16	COBOL 0S/390	
.	BIGVAR	2003/02/26	14.31.11	15	COBOL 0S/390	
.	BLLSUB	2002/09/09	18.38.02	11	COBOL II	
.	BLLTEST	2002/09/09	17.05.07	11	COBOL II	
.	BLVTEST	2002/09/26	14.25.35	16	COBOL 0S/390	
.	CALLDETH	2003/10/06	14.33.21	11	COBOL II	
.	CAMRTPIP	2003/08/22	15.52.00	21	HLASM	
.	CAOUMMAN	2003/12/09	12.50.00	43	HLASM	
.	CA02DEMO	2004/03/12	16.35.54	90	COBOL MVS	CA-OPT
.	COMPODO	2003/09/04	15.46.56	32	COBOL II	
.	DFSLET1	2003/10/08	16.39.00	10	HLASM	
.	DISASMT	2003/11/05	17.07.00	26	HLASM	
.	FSSC10	2004/03/03	17.37.14	64	COBOL II	
.	IDECLEAR	2002/08/28	15.15.44	42	COBOL 0S/390	

The PROTSYM (or CSL) Directory panel displays an entry for each available member in your symbolic file.

PROTSYM Members

For PROTSYM members, the following information is displayed:

- **Date and Time**
Displays the date and time when the program member was compiled.
- **Size**
Displays the number of PROTSYM records occupied by the member.
- **Language**
Displays the compiler used.
- **Attributes**
Indicates whether a program was CA-optimized or loaded into the PROTSYM with the NOPURGE option.

CSL Members

For CSL members, the following information is displayed:

- **Date and Time**
Displays the date and time when the program member was compiled.
- **Genlevel**
Displays the maintenance level of the Optimizer or postprocessor used to create the member.
- **Release**
Displays the release number of the Optimizer or postprocessor used to create the member.
- **Compiler**
Displays the compiler used.

Delete a Member from a Symbolic File

Follow these steps:

1. Select SYM option 1 to display the PROTSYM Directory or CSL Directory panel.

2. Type the D line command to delete the desired member.

Because this action is potentially destructive, you are prompted for confirmation before the member is deleted, as shown in the following screen:

```
CA SymDump Batch ----- PROTSYM Directory ----- Line 1 of 42
Command ==> Scroll ==> CSR
-----
PROTSYM Ds +-----+
-----+-----+
Cmd  Progr | Press ENTER to confirm delete for member ATTNTEST | ----- Lvl 2
      | or END to cancel the delete request.           | butes
      | Display this message? (Y/N) ==> Y              |
      |-----+-----+
      | BIGSE |
      |-----+-----+
      | BIGVAR 2003/02/26 14.31.11 15 COBOL OS/390
      | BLLSUB 2002/09/09 18.38.02 11 COBOL II
      | BLLTEST 2002/09/09 17.05.07 11 COBOL II
      | BLVTEST 2002/09/26 14.25.35 16 COBOL OS/390
      | CALLDETH 2003/10/06 14.33.21 11 COBOL II
      | CAMRTPIP 2003/08/22 15.52.00 21 HLASM
      | CAOUMMAN 2003/12/09 12.50.00 43 HLASM
      | CA02DEMO 2004/03/12 16.35.54 90 COBOL MVS CA-OPT
      | COMPODO 2003/09/04 15.46.56 32 COBOL II
      | DFSLET1 2003/10/08 16.39.00 10 HLASM
      | DISASMT 2003/11/05 17.07.00 26 HLASM
      | FSSC10 2004/03/03 17.37.14 64 COBOL II
      | IDECLARE 2002/08/28 15.15.44 42 COBOL OS/390
```

3. Press Enter to delete the member, or END to cancel the request.

To suppress confirmation for the remainder of the session, type **N** in the Display this message? field located at the bottom of the confirmation pop-up window.

To suppress all delete confirmations for your ID, use the PROFILE command.



Note: If you are using passwords (SYMPSWD=password), a password must be entered before a member can be deleted from a PROTSYM file. The password is established during product installation. Check with your system administrator and obtain the password before attempting to delete a PROTSYM member. Use the PROFILE command to store the password in your profile so you do not need to enter it every time.

If you are using an external security interface (EXTSEC=ALL/SYMBOLIC), verify that you have delete authority and that the security functions have been defined.

For more information, see [Global Options \(https://docops.ca.com/display/CAITSD11/Options\)](https://docops.ca.com/display/CAITSD11/Options).

Print a Program Listing from a Symbolic File

Follow these steps:

1. Select SYM option 1 to display the PROTSYM Directory or CSL Directory panel.
2. Type the P line command to print the desired program listing.
You are prompted for the destination as shown in the following screen:

```
CA SymDump Batch ----- PROTSYM Directory ----- Line 1 of 42
Command ==> Scroll ==> CSR
-----+-----+-----+
PROT +-----+-----+-----+ Lvl 2
----|-----|-----|-----|
Cmd | Print Options: * Press ENTER to PRINT *
P   |
.   | Printer      ==> USILDAVC.USER01
.   | or
.   | Dsname       ==> 'USER01.APSDN'
.   | Member       ==> NEW          Disposition ==> NEW
.   |
.   | Copies       ==> 001
.   | Class        ==> X
.   | PageDef      ==>
.   | FormDef      ==>
.   | Chars        ==>
.   |
.   |
.   |
.   |
.   |
.   |
```

3. Fill in the following fields and press Enter to print the member listing:

- **Printer**

Lets you specify the destination to which the printed output is routed. This can be specified as a printer name, node.userid, or blank to spool the printed output to your own TSO userid.

- **Dsname**

Lets you define the name of a data set to which the printed output is written. If this field is non-blank, it overrides the Printer field. When entering a data set name, it must be fully qualified and enclosed in single quotation marks. Otherwise, your ZPREFIX value is used as the high-level qualifier.

- **Member**

Lets you specify the member name if you are printing the listing to a partitioned data set.

- **Disposition**

Lets you specify the disposition of the data set to which the output should be printed. This field is relevant only if printing to a data set. If the value is NEW or MOD, a second window pops up to request allocation parameters.

Values: SHR, OLD, NEW, and MOD



Note: For a description of the fields on the window, see the [PRINT](#) (see page 299) command.

- **Copies**

Lets you specify the number of copies to be printed.

- **Class**

Lets you specify the sysout class that is used to allocate the print data set.

- **PageDef**

Lets you specify the name of the library member that PSF uses to define the page layout for printing on a 3800 Printing Subsystem Model 3.

- **FormDef**

Lets you specify the name of the library member that PSF uses to define the form layout for printing on a 3800 Printing Subsystem Model 3.

- **Chars**

Lets you specify the name of the character arrangement tables for printing on a 3800 Printing Subsystem Model 3.

View a Program Listing from a Symbolic File

Follow these steps:

1. Select SYM option 1 to display the PROTSYM Directory or CSL Directory panel.
2. Type the S line command to select the desired listing for viewing.

```
CA SymDump Batch ----- CA02DEMO ----- Line 620 of 2714
Command ==>                               Scroll ==> CSR
-----
PROTSYM Dsname: CAI.PROTSYM
-----
.....1.....2.....3.....4.....5.....6.....7..... Lvl 3
000478 *****
000479 **          P R O C E D U R E   D I V I S I O N
000480 *****
000481 *****
000482 *****
000483 MAIN-PROGRAM-LOGIC SECTION.
000484
000485     PERFORM PRE-INIT.
000486     PERFORM INITIALIZE-DEMO.
000487
000488 MAIN-REPEAT-LOOP SECTION.
000489     IF COUNT-TIMES LESS THAN 1 GO TO COMPLETE-DEMO.
000490     SUBTRACT 1 FROM COUNT-TIMES.
000491
000492     PERFORM SECTION-A.
000493     PERFORM SECTION-B.
000494     PERFORM SECTION-C.
000495     PERFORM SECTION-D.
000496     PERFORM SECTION-E.
000497     PERFORM SECTION-F.
000498     PERFORM SECTION-G.
000499     PERFORM SECTION-H.
000500     PERFORM SECTION-I.
000501     PERFORM SECTION-J.
000502     PERFORM SECTION-K.
000503     PERFORM SECTION-L.
000504     PERFORM SECTION-M.
000505     PERFORM SECTION-N.
000506     PERFORM SECTION-O.
000507     PERFORM MAIN-MLT-PARAGRAPH
000508         THRU MAIN-MLT-PARAGRAPH-EXIT.
```

The static information area at the top of the display continues to show the data set name of the selected symbolic file.

If your Keep Window is active and contains data, it continues to display.

The final static line in this display is the highlighted ruler, which scrolls left and right automatically as you scroll the listing.

The data portion of the display is both vertically and horizontally scrollable when the report length or width cannot be fully accommodated by your screen dimensions.

Add a Listing to a PROTSYM File

Follow these steps:

1. Select SYM option 2 to add one or more listings to a PROTSYM file.

This option provides an online interface to the symbolic loader. It can be used to add a single listing, a range of members, or an entire library at one time. It supports program listings generated by any of the supported IBM compilers and residing in any of the supported library formats.

The Add Listing to Protsym pop-up window prompts you for update parameters, as shown in the following screen:

```

CA SymDump Batch ----- Report Index ----- Line 1 of 68
Command ==> Scroll ==> CSR
-----
Re +-----+
Fi | Add Listing to Protsym |
-- | Protym Dsname ==> 'CAI.PROTSYM' | l 1
Cm | Listing Dsname ==> 'USER01.LISTINGS' | d
  | Library Type ==> PDS (PDS, SEQ, PAN, LIB, NDV) | R7
  | From Member ==> COB2DEMO | R3
  | To Member ==> | R1
  | View Messages ==> ALL (ALL, NONE, RC) | R7
  +-----+
  | WANDA07C TESTCOB2 SUBCOB 000026A S=0C7 2007/03/02 16.25 TSTUSR7 |
  | WANDA07A S0C1 RENTSCAN 0000AFA S=0C2 2007/03/01 11.00 TSTUSR7 |
  | USER002A RUN CAOEDMO 0003D48 S=0C7 2007/03/01 10.41 TSTUSR3 |
  | USER002A RUN CAOEDMO 0003D48 S=0C7 2007/03/01 10.34 TSTUSR7 |
  | USER002A RUN CAOEDMO 0003D48 S=0C7 2007/02/28 10.06 USER01 |
  | WANDA07Z S0C1 TESTSCAN 0000AFA S=0C2 2007/02/28 09.43 USER01 |
  | USRSC01B RUN CARXDEMA 000013C S=0C7 2007/02/23 14.34 USER02 |
  | USRSC01A RUN CEEPLPKA 00BAE7A U=0666 2007/02/23 14.33 USER02 |
  | USRSC01A STEP1 CARXDEMA 000013C S=0C7 2007/02/23 14.28 USER02 |
  | USRSC01A RUN ABEND1 0000014 U=0001 2007/02/21 15.26 TSTUSR2 |
  +-----+

```

2. Fill in the following fields and press Enter to update the PROTSYM file:

- **PROTSYM Dsname**

Lets you specify the data set name of any PROTSYM file. When fully qualified, you must type the name in single quotation marks. Otherwise, the value of ZPREFIX is added as the high-level qualifier.

- **Listing Dsname**

Lets you specify the data set name of your listing data set or library. When fully qualified, you must type the name in single quotation marks. Otherwise, the value of ZPREFIX is added as the high-level qualifier.

- **Library Type**

Lets you specify one of the following supported listing library types:

- **PDS**

Partitioned data sets

- **SEQ**

Sequential listing file

- **PAN**

CA Panvalet library

- **LIB**

CA Librarian library

- **NDV**

CA Endeavor SCM library

- **From Member**

Lets you add the member names. To add a single member, type the member name in this field. For multiple members whose names share a common prefix, type the prefix in this field with a single trailing asterisk (*) to indicate wild-carding. To add a range of members, type the first member of the range.

- **To Member**

Lets you specify a range of members, To add a range of members, type the name of the last member in this field. Otherwise, leave this field blank.

- **View Messages**

Lets you view messages using the following options:

- ALL -- Shows all messages generated while the listings are being added to the PROTSYM file.
- RC -- Shows a single message for each member containing the highest return code produced while adding the member.
- NONE -- Suppresses all messages.

If you have requested to view output messages from the utility, the output appears automatically after the utility is completed, as shown in the following screen:

```
CA SymDump Batch ----- Add to PROTSYM ----- Line 1 of 28
Command ==>                                     Scroll ==> CSR
-----
PROTSYM Dsname: CAI.PROTSYM                      From: COB2DEMO
Listing Dsname: USER01.LISTINGS                  To:
----- Lvl 2
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
COB2DEMO POST-PROCESSING ENDED, IN25COB2 RETURNED 00000000

IN25COB2 6.2 - 02/18/2004 10.25

SYM036 PASSED PARAMETER STATEMENTS

COB2DEMO, LISTER=ALL, CUTPRINT=ALL

SYM010 PROCESSING HAS BEGUN FOR PROGRAM - COB2DEMO

SYM006 LISTER=ALL      REQUESTED
SYM007 CUTPRINT=ALL    REQUESTED
```

NESTED PROGRAMS	ID	WS	LS	PD	EP
COB2DEMO	000003	000023	000864	000875	000000

SYM024 INPUT FILE PROCESSED

SYM020 SYMBOLIC FILE UPDATED SUCCESSFULLY

SYM021 117 SOURCE STATEMENTS SAVED

SYM022 143 TOTAL RECORDS INSERTED INTO SYMBOLIC FILE

SYM023 POST-PROCESSOR TERMINATED

3. Press END to return to the Add Listing to PROTSYM pop-up window.

List Globally Defined Symbolic Files

Select SYM option 3 to list the names of the symbolic files which are globally defined in your installation defaults member, CAOUDFRX. These symbolic files are automatically available for report formatting by any user or job.

```
CA SymDump Batch----- Global Symbolic Files ----- Line 1 of 5
Command ==>>>                                     Scroll ==>>> CSR
----- Lvl 2
Cmd  Symbolic File Dsname                                Type
.    'AD1DEV.SB21.PROD.PROTSYM'                          Protsym
.    'AD1DEV.SB21.DEVL.PROTSYM'                          Protsym
.    'AD1DEV.UTILITY.RROTSYM'                            Protsym
.    'AD1DEV.SB20.PROD.CSL'                              CSL
.    'AD1DEV.UTILITY.CSL'                                CSL
-----
```

Globally defined symbolic files can be PROTSYM or CSL format and there is no limit to the number of files that can be defined for your installation.

The following additional functions can be performed by entering line commands in the Cmd column on this panel:

- **A**
Adds listings to this symbolic file. This command displays the Add Listing To PROTSYM pop-up window. Only listings can be added to PROTSYM files.
- **S**
Selects a symbolic file for viewing. This command displays the PROTSYM Directory or CSL Directory panel, depending on the file format.

To globally define additional symbolic files, contact your system administrator.

List Supplemental Symbolic Files

Select SYM option 4 to list the names of the symbolic files which you have locally defined for your use. This option lets you maintain and use your own personal symbolic files for your own testing environments without affecting other users.

Your locally defined supplemental symbolic files are stored in your user profile.

```
CA SymDump Batch----- Supplemental Symbolic Files ---- Line 1 of 16
Command ==>>>                                     Scroll ==>>> CSR
```

----- Lvl 2		
Cmd	Symbolic File Dsname	Type
.	'USRSC01.TEST.PROTSYM'	Protsym
.	'AD1DEV.MIKED.PROTSYM'	Protsym
.	'CDE.DEVL.IB31.PROTSYM'	Protsym
.	'CDE.PROD.IB31.PROTSYM'	Protsym
.		
.		
.		
.		
.		
.	'MIKED.QA.CSL'	CSL
.	'USRSC01.CSL'	CSL
.	'USER002.CSL'	CSL
.		
.		
.		

Locally defined supplemental symbolic files can be PROTSYM or CSL format and you can have up to 16 supplemental files defined at any one time.

To add a new file, type the data set name on an empty line. When fully qualified, you must type the name in single quotation marks. Otherwise, the value of ZPREFIX is added as the high-level qualifier.

To remove a supplemental symbolic file, clear the data set name from the line.

The following additional functions can be performed by entering line commands in the Cmd column on this panel:

- **A**
Adds listings to this symbolic file. This command displays the Add Listing To Protsym pop-up window. Only listings can be added to PROTSYM files.
- **S**
Selects a symbolic file for viewing. This command displays the PROTSYM Directory or CSL Directory panel, depending on the file format.

View Dynamic Symbolic Support Options

Use the Dynamic Symbolic Support\ Options pop-up to view dynamic symbolic support options for the Repository Viewer.

To view the dynamic symbolic support options, select option 5 from the Symbolic Utilities menu.

The Dynamic Symbolic Support\ Options pop-up opens.

CA SymDump Batch ----- Report Index -----		Line 1 of 68
Command ==>		Scroll ==> CSR

Re	Dynamic Symbolic Support Options	* Press ENTER to Save *
Fi		
--	NDVRDSS ==> Y (Y/N/) Activate Dynamic Symbolic Support	1 1
Cm	NDVRASM ==> Y (Y/N/) Always Populate non LE-enabled Assembler	d
.	NDVRDSN ==> 'USER07.PROTSYM'	7
.		7
.	To use the option value in effect at the time a report was	7
.	created, leave it blank.	7
.		7

```

. +-----+ 7
. WANDA07A FDCHK CEEPLPKA 00C3A0C U=4038 2007/09/12 14.12 USER07
. WANDA07A S0C1 LEASM0C1 00000F2 S=0C1 2007/09/12 14.12 USER07
. WANDA07A CAMRCOB2 SUBCOB2 000035E S=0C7 2007/09/12 14.11 USER07
. WANDA07B CAMRPLI CAMRPLI1 00006F2 S=0C7 2007/09/12 14.07 USER07
. USER002R RUN LEASM0C1 00000F2 S=0C1 2007/09/12 13.47 USER02
. WANDA07B TESTBEAR UNKWN1 0000000 S=0C1 2007/09/12 13.00 USER07
. WANDA07B TESTBEAR UNKWN1 0000000 S=0C1 2007/09/12 12.40 USER07
. WANDA07B CAMRPLI CAMRPLI1 00006F2 S=0C7 2007/09/12 12.38 USER07
. WANDA07B TESTBEAR UNKWN1 0000000 S=0C1 2007/09/12 10.26 USER07
. WANDA07B CAMRPLI CAMRPLI1 00006F2 S=0C7 2007/09/11 16.16 USER07
. WANDA07B CAMRPLI CAMRPLI1 00006F2 S=0C7 2007/09/11 16.12 USER07
. WANDA07A TESTBEAR UNKWN1 0000000 S=0C1 2007/09/11 16.04 USER07
. USER002R RUN @BMPERRI 000068E U=4038 2007/09/10 13.59 USER02
. USRSC01A STEP1 T64 00000D2 S=0C7 2007/09/07 13.37 USER01

```

Override Dynamic Symbolic Support Options

Use the Dynamic Symbolic Support\ Options pop-up to change dynamic symbolic support options for the Repository Viewer.

Follow these steps:

1. View the Dynamic Symbolic Support\ Options pop-up as described in the previous procedure.
2. Update the following fields and press Enter.

▪ NDVRDSS

Lets you specify whether to activate or deactivate dynamic symbolic support for CA Endeavor SCM.

Values:

- **Y** -- Activates dynamic symbolic support for your ISPF session.
- **N** -- Deactivates dynamic symbolic support for your ISPF session.
- **Blank** -- Uses the option value in effect for each report at the time the report was created.

▪ NDVRASM

Lets you specify whether to auto-populate the symbolic file for non-LE enabled Assembler programs when viewing reports in your ISPF session. This option takes effect only when the NDVRDSS option is set to ON.

Values:

- **Y** -- Always auto-populates symbolic files for non-LE-enabled Assembler programs when viewing reports during your ISPF session.
- **N** -- Skips auto-populate of symbolic files for non-LE-enabled Assembler programs when viewing reports during your ISPF session.
- **Blank** -- Uses the option value in effect for each report at the time the report was created.

▪ NDVRDSN

Lets you specify the data set name of a PROTSYM file designated as the receiver of auto-populated symbolic files.

When the data set name is fully qualified, enclose the name in quotes. If you do not use quotes, the value of *ZPREFIX* is added as the leading qualifier.
 Leave the NDVRDSN field blank to use the DSN in effect for each report at the time the report was created.



Notes:

- When you change one or more fields before pressing Enter, the fields are validated and redisplayed. To continue, press Enter one more time.
- The option values specified on this screen are locally defined and are only used when formatting a report using the Repository Viewer in your ISPF session.
- The NDVRDSN value specified here is concatenated with your other symbolic files. In addition, if you specified an NDVRDSN value (or CAINDVR DD) at execution time, that data set is included in the search list as well.

Repository Viewer Commands

This section contains an alphabetical listing of the CAIPRINT Repository Viewer primary commands.

- [FIND \(see page 294\)](#)
- [FM \(see page 296\)](#)
- [HELP \(see page 296\)](#)
- [KEEP \(see page 297\)](#)
- [LOCATE \(see page 298\)](#)
- [NOTES \(see page 298\)](#)
- [OPTIONS \(see page 299\)](#)
- [PRINT \(see page 299\)](#)
- [PROFILE \(see page 301\)](#)
- [REFRESH \(see page 303\)](#)
- [RFIND \(see page 303\)](#)
- [SETINDEX \(see page 304\)](#)
- [SORT \(see page 305\)](#)
- [SYM \(see page 305\)](#)
- [TAG \(see page 306\)](#)
- [VIEW \(see page 306\)](#)

FIND

Use the FIND command to locate a string in the current display.

Syntax

Find <string> [P|F|L] [col1 [col2]]

Synonyms

None.

Parameters

- **string**
Lets you specify a 1- to 32-character text string (including quotes) that you can use as a search argument. If it contains blanks or commas, it must be enclosed in quotes.
- **P**
Lets you locate the string by searching backward from the current cursor position.
Values:
P or PREV
- **F**
Lets you locate the first occurrence of the string by searching forward from the top of the data.
Values:
F or FIRST
- **L**
Lets you locate the last occurrence of the string by searching backward from the bottom of the data.
Values:
L or LAST
- **col1**
Lets you specify a 1-origin column number in which the string must begin, or the 1-origin column number that begins the range of columns in which the entire string must be found.
- **col2**
Lets you specify a 1-origin column number that ends the range of columns in which the entire string must be found.

Usage Notes:

- If the string is found, the cursor is positioned to the start of the string.
- If the string was found on a line or in a column that was not previously in view, the display is scrolled just enough to display the string.
- Use the RFIND command, or FIND without an argument, to repeat the previous search from the current cursor position.
- If the previous search reached the top or bottom of the file without successfully locating the search argument, a repeat find command starts searching from the other end of the display, but always in the same direction as the previous search.
- Following a successful FIND command using the FIRST or LAST options, RFIND can be used to locate the next or previous occurrence, respectively.
- Searching for non-displayable data is not supported.

FM

Use the FM command to view the contents of a data set using CA File Master Plus.

Syntax

FM <data-set-name>

Synonyms

None

Parameters

- **data-set-name**
Data-set-name lets you specify the data set whose contents you would like to view using CA File Master Plus.

Usage notes:

- If no argument is specified, the FM command is sensitive to the current cursor position. To view the contents of a data set whose name is currently displayed in the data area of the screen, place your cursor on the data set name and press Enter.
- You must have CA File Master Plus installed and customized for this command to work. For more information, see [Configuring \(https://docops.ca.com/display/CAITSD11/Configuring\)](https://docops.ca.com/display/CAITSD11/Configuring) and CA File Master Plus documentation.

HELP

Use the HELP command to display information about any command, message, topic, or panel.

Syntax

Help <arg | COMMANDS | MESSAGES | SCREENS | TOPICS | ALL>

Synonyms

None.

Parameters

- **arg**
Lets you specify the name of any Repository Viewer command, message, panel, or other help topic. For example, to request help for the PRINT primary command, type **HELP PRINT** on the command line and press Enter.
- **COMMANDS**
Lets you request a list of all Repository Viewer commands, from which you can select any command to request HELP.
- **MESSAGES**
Lets you request a list of all Repository Viewer messages, from which you can select any message to request HELP.

- **SCREENS**

Lets you request a list of all Repository Viewer panels, including pop-up windows, from which you can select any panel to request HELP.

- **TOPICS**

Lets you request a list of all Repository Viewer help topics other than commands, messages, or panels, from which you can select any topic to request HELP.

- **ALL**

Lets you request a list of all Repository Viewer help topics, from which you can select any topic to request HELP.

Usage Notes:

- To request HELP for any Repository Viewer message currently displayed on your panel, no argument is required. Simply press the PF key you have set for HELP, or type **HELP** on the command line and press Enter.
- If no message is currently displayed on your panel, entering a HELP command without any argument produces the help topic for the current panel or pop-up window.

KEEP

Use the KEEP command to add a data line from the current display to the Keep Window, to remove a previously added data line from the Keep Window, or to change the display status of the Keep Window.

Syntax

Keep [ON|OFF]

Synonyms

None.

Parameters

- **ON**

Lets you enable the Keep Window display, if the window is not empty.

- **OFF**

Lets you disable the Keep Window display, suppressing the contents without discarding it.

Usage Notes:

- If no argument is specified, the KEEP command is sensitive to the current cursor position. When the cursor is positioned on a data line within the display, that line is added to the Keep Window. When the cursor is positioned on a line within the Keep Window, that line is removed from the Keep Window.
- Data cannot be added or removed from the Keep Window when the window is disabled.

LOCATE

Use the LOCATE command while viewing an abend report to position your display to a previously established tag.

Syntax

Locate <name>

Synonyms

None.

Parameters

- **name**
Lets you specify the 1- to 32-character name of the previously defined tag, which will be used to reposition your display.

Usage Notes:

None.

NOTES

Use the NOTES command to open the Electronic Notepad for the current report.

Syntax

Notes

Synonyms

None.

Parameters

None.

Usage Notes:

- If a notepad does not exist for a report, you are prompted for confirmation before a new notepad is created.
- Notes are stored in your profile, which is keyed by abending program name, abending program offset, and completion code. Therefore, two or more reports which share these attributes will also share a notepad.
- You can launch the electronic notepad for any report from the Report Index panel using the n line command.

OPTIONS

Use the OPTIONS command to view a report of your system-wide installation options, including component genlevels and current status.

Syntax

Options

Synonyms

Opts

Parameters

None.

Usage Notes

The same report can be obtained in batch by executing program CAOUOPTS using the sample JCL provided in CAI.CAVHJCL.

PRINT

Use the PRINT command at any time to print all or part of the current display to a printer, node.userid destination, your TSO userid, or a data set.

When you type a PRINT command, your print options appear in the Print Options pop-up window.

Modify the following values and press Enter to print, or press END to cancel the request.

Syntax

PRint

Synonyms

None.

Parameters

The PRINT command prompts you for input parameters using the Print Options pop-up window with the following fields:

- **Printer**

Lets you specify the destination to which the printed output should be routed. This can be specified as a printer name, node.userid, or blank to spool the printed output to your TSO userid.

- **Dsname**

Lets you specify the name of a data set to which the printed output is written. If this field is non-blank, it overrides the Printer field. When entering a data set name, it must be fully qualified and enclosed in single quotation marks. Otherwise, your ZPREFIX value is used as the high-level qualifier.

- **Member**
Lets you specify the member name if you are printing the listing to a partitioned data set.
- **Disposition**
Lets you specify the disposition of the data set to which the output should be printed. This field is relevant only if printing to a data set. If the value is NEW or MOD, a second window pops up to request allocation parameters.
Values: SHR, OLD, NEW, and MOD
- **Title**
Lets you specify the 1- to 32-character title that appears at the top each page.
- **Print Lines**
Lets you specify the starting and ending lines to be printed. By default the starting line is set to one and the ending line is set to the number of lines in the report.
- **Copies**
Lets you specify the number of copies to be printed.
- **Class**
Lets you specify the sysout class that will be used to allocate the print output data set.
- **Lines / Page**
Lets you specify the number of logical lines per page, including the title and header lines.
- **Record Size**
Lets you specify the logical record length that will be used for the print data set. By default this is the maximum width of the data being printed. You can increase or decrease this value to fit your printer as needed. Decreasing the record size may result in truncation of data.
- **PageDef**
Lets you specify the name of the library member that PSF uses to define the page layout for printing on a 3800 Printing Subsystem Model 3.
- **FormDef**
Lets you specify the name of the library member that PSF uses to define the form layout for printing on a 3800 Printing Subsystem Model 3.
- **Chars**
Lets you specify the name of the character arrangement tables for printing on a 3800 Printing Subsystem Model 3.
- **Using a Disposition of New or MOD**
Lets you specify the disposition to be used. When a disposition of NEW or MOD is used, a second pop-up panel is displayed.

Modify the following values and press Enter to allocate the file and print, or press End to cancel the request.

- **Space**
Lets you specify the space unit for allocating the data set. Type TRK to allocate in tracks, CYL to allocate in cylinders, or BLK to allocate in blocks.

- **Primary**
Lets you specify the primary space quantity. Type a number from 1 to 999.
- **Secondary**
Lets you specify the secondary space quantity. Type a number from 0 to 999.
- **Directory**
Lets you specify the number of blocks to be contained in the directory of the partitioned data set. If no member was specified when the data set was entered, this field is not updatable.
- **Dataclas**
Lets you specify the data class of an SMS-managed data set.
- **StorClas**
Lets you specify the storage class of an SMS-managed data set.
- **MgmtClas**
Lets you specify the management class of an SMS-managed data set.
- **Blkize**
Lets you specify the block size of the data set.
- **LRecl**
Lets you specify the logical record length of the data set. This field is not updatable.
- **Unit**
Lets you specify the Unit specification of the data set.
- **Volser**
Lets you specify the volume serial on which the data set will reside. If creating a new data set using DISP=MOD, this field must be left blank.

Usage Notes:

- When one or more fields are changed before pressing Enter, the fields are validated and redisplayed along with the message Press Enter to Print. To continue, press Enter one more time.
- At the top of each printed page, the page header appears. The header contains your specified title, centered on the line, and the page number at the far right.
- On the first page only, following the header line, the information area is printed with double-space carriage control.
- On every page, the contents of the panel header line is printed next, also with double-space carriage control. This may consist of column headings for a member list, report index or report tree display, or a ruler for listings and abend report contents.

PROFILE

Use the PROFILE command to view or update your personal viewer preferences.

Syntax

PRofile

Synonyms

None.

Parameters

The PROFILE command displays the Profile Settings pop-up window. You are prompted for input parameters using the following fields:

▪ Confirm Exit from Reports

Lets you specify whether to request a confirmation pop-up window or to suppress the information when you close an open report.

Values:

- Y -- Requests a confirmation pop-up window whenever you attempt to exit an open report.
- N -- Suppresses the confirmation.

▪ Confirm Report Deletes

Lets you specify whether to request a confirmation pop-up window or to suppress the confirmation when you attempt to delete a report.

Values:

- Y -- Requests a confirmation pop-up window when you attempt to delete a report.
- N -- Suppresses the confirmation.

▪ Confirm Sym Member Deletes

Lets you specify whether to request a confirmation pop-up window or to suppress the confirmation when you attempt to delete a member from a symbolic file.

Values:

- Y -- Requests a confirmation pop-up window.
- N -- Suppresses the confirmation.

▪ Force Upper Case Viewer

Lets you specify whether to force all output from the Repository Viewer, including output from the PRINT command, to be displayed entirely in uppercase or to enable mixed-case displays and printing.

Values:

- Y -- Displays all output from the Repository Viewer entirely in uppercase.
- N -- Enables mixed-case displays and printing.

▪ Display Information Area

Lets you specify whether to let the information area display immediately below the command line in every viewer display or to suppress the information area on all displays.

Values:

- Y -- Enables the information area to display immediately below the command line in every viewer display.

- N -- Suppresses the information area on all displays.

Note: Filters are part of the information area. If this field is set to N, the filters will not be displayed. However, their values will still affect the Report Index selection list.

- **Message Alarm Severity**

Lets you specify the lowest message severity I, W, or E, which should trigger the alarm to sound when messages of equal or greater severity are displayed on your panel.

- **PROTSYM Password (Hidden)**

Lets you specify the one- to eight-character password for your PROTSYM files that was created by your system administrator when this product was installed. If omitted, you are prompted for a PROTSYM file in each session.

Usage Notes:

- Press END to cancel the update request, or Enter to save your changes.
- When one or more fields are changed before pressing Enter, the fields are validated and redisplayed. To continue, press Enter one more time.
- Profile changes take effect immediately.

REFRESH

Use the REFRESH command from the Report Index panel to refresh the contents of the current display.

This has the same effect as using a SETINDEX command to reload the repository index without changing the repository data set name.

Syntax

REFresh

Synonyms

None.

Parameters

None.

Usage Notes:

None.

RFIND

Use the RFIND command to repeat a previous search.

Syntax

RFind

Synonyms

None.

Parameters

None.

Usage Notes:

- Using RFIND produces exactly the same results as using FIND without a search argument.
- If no previous FIND existed, the RFIND command ends in error.
- If the string was found on a line or in a column that was not previously in view, the display is scrolled just enough to display the string.
- If the previous search reached the top or bottom of the file without successfully locating the search argument, a repeat FIND command starts searching from the other end of the display, but always in the same direction as the previous search.
- Searching for non-displayable data is not supported.

SETINDEX

Use the SETINDEX command from the Report Index panel to switch your view to another repository data set.

Syntax

SETIndex <dsname>

Synonyms

None.

Parameters

- **dsname**
Lets you specify the data set name of the new repository file. If the data set name is enclosed in single quotation marks, it is fully qualified. Else, your ZPREFIX value is used for the leading qualifier.
If the data set name is specified, this value is used to populate the dsname field in the Enter Repository Dsname pop-up window that displays when a SETINDEX command is entered.

Usage Notes:

- If you enter an invalid repository name, or the repository contains no abend reports, you are presented with a blank index display. Repeat the command using a valid data set name.
- To select the installation default repository data set for your system, do not enter a data set name when prompted.

- The repository data set name is stored in your personal profile and retrieved again the next time you start the viewer.

SORT

Use the SORT command to sort the current display in ascending or descending order on any column header. The SORT command is only valid from the following displays:

- Report Index panel
- Symbolic File Member select list panel

Syntax

Sort <column-header> [A|D] [H]

Synonyms

None.

Parameters

- **column-header**
Lets you specify the name of the column on which the sort should be performed, from the header line at the top of the display.
- **A|D**
Lets you specify the values A or D to sort the display in ascending or descending order.
Values: A, D
- **H**
Lets you sort the hexadecimal numeric values. This allows digits A - F to compare greater than 0 - 9.

Usage Notes:

After sorting a display, SORT * returns the display to its original order.

SYM

Use the SYM command to display the Symbolic Utilities menu.

Syntax

SYM <option>

Synonyms

CSL, CPP, LISTcsl, PROTsyl, Utilities, DSS

Parameters

- **option**
Lets you prime the selection field to any of the valid menu items:

- List the contents of a PROTSYM or CSL symbolic file. After the list is displayed, you can browse, print, or delete members using line commands from the selection list.
- Add listings to a PROTSYM file.
- List symbolic files defined globally in your installation options table CAOUDFRX. After the list is displayed, use the S line command to list the contents of any symbolic file, or the A line command to add members to a PROTSYM file.
- Add or Remove supplemental symbolic files defined locally to your viewer profile. After the list is displayed, use the S line command to list the contents of any symbolic file, or the A line command to add members to a PROTSYM file.
- View or change the CA Endeavor SCM symbolic support options defined locally to your viewer profile. Once the options display, overtype the values to change them.

Usage Notes:

- The last symbolic file referenced by any of the symbolic utilities is automatically saved in the Symbolic File Dsname field.
- For online help for a symbolic utility, select the option for that utility and press PF1.

TAG

Use the TAG command while viewing a report to equate a line number to your current position within the report.

Syntax

TAG <name>

Synonyms

.name

Parameters

- **name**
Lets you specify any 1- to 32-character name that will become equated with your current position within the report you are viewing.

Usage Notes:

- Entering a duplicate tag name replaces the previous tag with the new report position. Two tags cannot share the same name.
- Use the LOCATE command to position your display using a previously defined tag.
- All tags are automatically deleted when you close and exit a report.

VIEW

Use the VIEW command while viewing a report to display the listing for a specified program.

The VIEW command displays the version of the listing that was used to provide symbolic information when the report was formatted. If no symbolic information was provided for the program in the report you are viewing, use the SYM command instead.

Syntax

View <program-name>

Synonyms

None.

Parameters

- **program-name**
Lets you specify the program name that was formatted with symbolic information in the report you are viewing.

CA SymDump Batch Utilities

The following reporting facilities are provided to measure the use and effectiveness of the product:

- The Management Reporting System (MRS) provides a means for you to audit your load module libraries.
- The Initialization Summary generates a report describing which CA SymDump Batch modules are currently loaded in the CSA area of LPA.
- The Options Summary reports on the options set as installation defaults.
- The CSL Summary reports on the contents of a specified CSL or on all of the CSLs defined as installation defaults.
- The CAIPRINT Repository Utility provides a means for you to list and maintain the contents of the central VSAM repository.

Symbolic file (PROTSYM) utilities are also provided. For a description of these utilities, see [Symbolic Support \(https://docops.ca.com/display/CAITSD11/Symbolic+Support\)](https://docops.ca.com/display/CAITSD11/Symbolic+Support).

Management Reporting System (MRS)

The Management Reporting System scans any load module library and reports on the COBOL programs contained within each member. MRS produces two reports, a detailed report and a summary report, indicating the size and characteristics of each load module in a selected library.

MRS reads the members of a specified library and identifies and lists COBOL CSECTs and load modules that contain them. It also provides compilation and size information for these CSECTs and indicates whether they are processed by CA Optimizer or CA Optimizer/II.

- [JCL \(see page 308\)](#)

- [Reports \(see page 308\)](#)
- [Usage Considerations \(see page 312\)](#)

JCL

A procedure called CAIMRSII is provided to invoke MRS. To obtain detail and summary reports for library PROD1.LOADLIB, specify the following:

```
// EXEC CAIMRSII, LIBRARY='PROD1.LOADLIB'
```

The MRS program, CAOUMRS, uses the following DD statements:

DD Statement	Description
STEPLIB	Defines the library containing MRS.
SYSPRINT	Defines the MRS report file.
SYSLIB	Defines the library to be analyzed.



Note: The ddname for the library to be analyzed is an installation option with a default of SYSLIB. If the ddname was changed during installation, ensure that the procedure CAIMRSII was changed too.

Reports

MRS provides the following two reports for each load library that is scanned:

- The Load Library Analysis Detailed Report is a detailed listing of all COBOL CSECTs in the library.
- The Load Library Analysis Summary Report is a summary of all of the library's statistics.

Load Library Analysis Detailed Report

The Load Library Analysis Detailed report lists all of the load modules in the library that contain COBOL CSECTs. A sample report is shown next:

LEGEND FOR OPTIMIZER OPTIONS:
D=DTECT C=XCOUNT X=XTIME P=PFLOW S=SUBRNGCK R=PARMCHK B=DBGSLEEP F=FDCHECK W=WSINIT I=
CICS E=EOS L=WSCLEAR H=PUSHPOP O=CASORT

MODULE ID	MODULE RE	DATE	SIZE	CSECT TIME	CSECT-CA-OPT	SIZE REL	PROC-DCXPSR	DATA-BFWIEL	SIZE HO	COMPILER-
IX101A	9,376	IX101A	8,276		3,826		4,450		C2	5668-958 3.2 08/13
/2003	13.54.45	NONE								
IX102A	12,440	IX102A	11,340		6,432		4,908		C2	5668-958 3.2 08/13
/2003	13.54.45	NONE								
IX103A	13,128	IX103A	12,028		6,952		5,076		C2	5668-958 3.2 08/13
/2003	13.54.46	NONE								
IX104A	12,880	IX104A	11,782		6,266		5,516		C2	5668-958 3.2 08/13
/2003	13.54.45	NONE								
IX105A	17,048	IX105A	15,950		9,128		6,822		C2	5668-958 3.2 08/13
/2003	13.54.46	NONE								
IX106A	22,824	IX106A	21,726		10,990		10,736		C2	5668-958 3.2 08/13
/2003	13.54.45	NONE								
IX107A	15,536	IX107A	14,440		8,762		5,678		C2	5668-958 3.2 08/13
/2003	13.54.46	NONE								

IX108A	20,560	IX108A	19,458	12,840	6,618	C2	5668-958	3.2	08/13
/2003	13.54.09	NONE							
IX109A	15,968	IX109A	14,868	7,070	7,798	C2	5668-958	3.2	08/13
/2003	13.54.15	NONE							
IX110A	10,048	IX110A	8,948	4,168	4,780	C2	5668-958	3.2	08/13
/2003	13.54.15	NONE							
IX111A	9,584	IX111A	8,486	3,454	5,032	C2	5668-958	3.2	08/13
/2003	13.54.15	NONE							
IX112A	14,224	IX112A	13,124	5,772	7,352	C2	5668-958	3.2	08/13
/2003	13.54.16	NONE							
IX113A	12,472	IX113A	11,372	4,206	7,166	C2	5668-958	3.2	08/13
/2003	13.54.16	NONE							
IX114A	11,944	IX114A	10,844	3,814	7,030	C2	5668-958	3.2	08/13
/2003	13.54.15	NONE							
IX115A	11,896	IX115A	10,796	3,770	7,026	C2	5668-958	3.2	08/13
/2003	13.54.49	NONE							
IX116A	11,888	IX116A	10,788	3,762	7,026	C2	5668-958	3.2	08/13
/2003	13.54.52	NONE							
IX117A	11,896	IX117A	10,796	3,766	7,030	C2	5668-958	3.2	08/13
/2003	13.54.51	NONE							
IX118A	11,928	IX118A	10,828	3,806	7,022	C2	5668-958	3.2	08/13
/2003	13.54.52	NONE							
IX119A	12,168	IX119A	11,068	3,976	7,092	C2	5668-958	3.2	08/13
/2003	13.54.51	NONE							
IX120A	11,704	IX120A	10,604	3,580	7,024	C2	5668-958	3.2	08/13
/2003	13.54.52	NONE							
IX121A	12,952	IX121A	11,852	4,660	7,192	C2	5668-958	3.2	08/13
/2003	13.54.52	NONE							
IX201A	9,424	IX201A	8,324	3,826	4,498	C2	5668-958	3.2	08/16
/2003	12.26.51	NONE							
IX202A	12,504	IX202A	11,404	6,468	4,936	C2	5668-958	3.2	08/16
/2003	12.26.52	NONE							
IX203A	13,128	IX203A	12,028	6,910	5,118	C2	5668-958	3.2	08/16
/2003	12.26.53	NONE							
IX204A	12,896	IX204A	11,798	6,298	5,500	C2	5668-958	3.2	08/16
/2003	12.26.53	NONE							
IX205A	19,728	IX205A	18,288	9,518	8,770	C2	5668-958	3.2	08/16
/2003	14.13.25	OPT/II 2.0 DCXPSR.FW....							
IX206A	13,904	IX206A	12,804	7,572	5,232	C2	5668-958	3.2	08/16
/2003	12.26.55	NONE							
IX207A	15,960	IX207A	14,864	8,760	6,104	C2	5668-958	3.2	08/16
/2003	12.26.56	NONE							
IX208A	24,776	IX208A	23,680	17,004	6,676	C2	5668-958	3.2	08/16
/2003	12.26.56	NONE							
IX209A	39,504	IX209A	38,404	26,034	12,370	C2	5668-958	3.2	08/16
/2003	12.26.56	NONE							
IX210A	33,616	IX210A	32,520	20,650	11,870	C2	5668-958	3.2	08/16
/2003	12.26.57	NONE							
IX211A	19,072	IX211A	17,972	8,240	9,732	C2	5668-958	3.2	08/16
/2003	12.26.57	NONE							
IX212A	18,936	IX212A	17,840	11,292	6,548	C2	5668-958	3.2	08/16
/2003	12.26.57	NONE							
IX213A	18,328	IX213A	17,232	11,240	5,992	C2	5668-958	3.2	08/16
/2003	12.26.57	NONE							
IX214A	34,032	IX214A	32,936	21,034	11,902	C2	5668-958	3.2	08/16
/2003	12.26.59	NONE							
IX215A	48,584	IX215A	47,486	33,168	14,318	C2	5668-958	3.2	08/16
/2003	12.27.01	NONE							
IX216A	13,400	IX216A	12,298	6,610	5,688	C2	5668-958	3.2	08/16
/2003	12.27.02	NONE							
IX217A	11,400	IX217A	10,302	5,168	5,134	C2	5668-958	3.2	08/16
/2003	12.27.01	NONE							
IX218A	10,648	IX218A	9,550	4,474	5,076	C2	5668-958	3.2	08/16
/2003	12.27.04	NONE							
IX301M	3,264	IX301M	2,164	694	1,470	C2	5668-958	3.2	08/16
/2003	12.27.08	NONE							
IX302M	3,896	IX302M	2,796	136	2,660	C2	5668-958	3.2	08/16
/2003	12.27.08	NONE							
IX401M	3,496	IX401M	2,398	464	1,934	C2	5668-958	3.2	08/16
/2003	12.27.08	NONE							
RL206A	13,256	RL206A	11,814	5,248	6,566	C2	5668-958	3.2	09/16

/2003	10.57.29	OPT/II	2.0	DCXPSR.FW....					
SQ124A	20,968	SQ124A		19,522	10,368	9,154	C2	5668-958	3.2 06/18
/2002	13.32.01	OPT/II	2.0	DCXPSR.FW....					
ST140A	22,232	ST140A		20,778	11,162	9,616	C2	5668-958	4.0 06/25
/2002	15.35.57	OPT/II	2.0	DCXPSR.FW....					
TOTALS:	719,416			668,576	363,338	305,238			

The report displays the following information:

▪ **LEGEND FOR OPTIMIZER OPTIONS**

Lets you specify the meaning of each letter that appears in the last column. All the options for execution-time facilities that can be specified at compile/optimize time are listed in the report.

▪ **MODULE**

Lists the names of the load modules that contain at least one COBOL CSECT.

▪ **MODULE-SIZE**

Displays the total number of bytes in each load module listed.



Note: For overlay modules, the size specifies the maximum amount of memory used by the module and not the sum of the CSECTs contained in it.

▪ **CSECT**

Lists all the COBOL CSECT names in each load module. Multiple COBOL CSECTs within a given load module print on a separate line.

▪ **CSECT-SIZE**

Displays the total number of bytes in each COBOL CSECT.

▪ **PROC-SIZE**

Displays the number of bytes for the program's Procedure code. See [Usage Considerations \(see page 312\)](#) for more information.

▪ **DATA-SIZE**

Displays the number of bytes of storage generated by the compiler for the COBOL program's Data Division. For more information about Data Division, see [Usage Considerations \(see page 312\)](#).

▪ **COMPILER-ID**

Identifies the type of COBOL compiler used and the compiler's program product number and version number. MRS copies the program product number and the version number directly from the COBOL module.

▪ **REL, DATE, and TIME**

Displays the release of the COBOL compiler used and the date and time of the program's compilation (copied directly from the COBOL module).

▪ **CA-OPT version release**

Verifies whether the COBOL CSECT is processed with a CA Optimizer product. The release number indicates the release with which the CSECT was optimized.

- YES indicates that the CSECT was optimized with an earlier release that did not retain release numbers in the load module.

- NONE indicates that the CSECT is not processed with CA Optimizer or CA Optimizer/II.

In some instances, the compiler program product number, version number, and the date and time of compilation may not be available. In this case, an UNAVAILABLE message appears. The last column shows whether certain CA Optimizer or CA Optimizer/II execution-time options were specified at compile or optimize time. If the program was optimized using CA Optimizer, then only the letters for DTECT, PFLOW, or XCOUNT appear even if the options FDCHECK, SUBRNGCK, or WSINIT were specified at compile time.

The last line of the report specifies the following totals in bytes (1024 KB):

- Load modules listed on the report
- COBOL CSECTs
- Number of data bytes
- Number of procedure bytes from the COBOL CSECTs

Load Library Analysis Summary Report

The Load Library Analysis Summary report follows the Load Library Analysis Detailed report and provides summarized statistical information about the library being scanned.

COMPILER-		ID	REL	MODULES	CSECTS	OPTOS	OPTII	DTE	XCO	XTI	PFL	SUB	PAR	DBG	FDC
	WSI		CIC	EOS	WSC	PUS	CAS								
C2 5668-															
958	3.2		44		44	0	3	3	3	3	3	3	0	3	
3	0		0	0	0	0									
C2 5668-															
958	4.0		1		1	0	1	1	1	1	1	1	0	1	
1	0		0	0	0	0									
TOTALS: COBOL															
	4		4	0	0	0	0	0	4	4	4	4	4	4	0
					45		45								
					45		45								
					4		4								
					45		94								

* For programs which have been compiled using CA-
OPTIMIZER, only DTECT, XCOUNT, and CICS are reflected in these MRS reports. *

* More information regarding the options used during optimization is only available fo
r programs compiled using CA-OPTIMIZER/II. *

For programs that are compiled using CA Optimizer, only DTECT, XCOUNT, and CICS are reflected in these MRS reports. More information regarding the options used during optimization is only available for programs compiled using CA Optimizer/II.

The Load Library Analysis Summary report displays the following information:

- **IGZ, CEE, CAO**
Displays the number of modules containing COBOL CSECTs and the total number of COBOL CSECTs.
- **ALL**
Displays the total number of modules and CSECTs in the library.

The next part of the Summary Report provides a summary of CA Optimizer and CA Optimizer/II options in effect by compiler type and version.

- **COMPILER ID**
Lists the compiler type and version used.
- **OPTOS**
Displays the number of programs compiled under the listed compiler version that were compiled and optimized with CA Optimizer.
- **OPTII**
Displays the number of programs compiled under the listed compiler version that were compiled and optimized with CA Optimizer/II.
- **TOTALS**
Displays the total number of programs that were optimized using the various options listed.

Usage Considerations

MRS analyzes any standard IBM load module library, but does not support concatenated load library data sets. Specify each load library to be analyzed in a separate MRS execution run.

MRS recognizes COBOL CSECTs that are compiled using the following IBM COBOL compilers: V2, V3, V4, OS/VS COBOL, COBOL II, COBOL 370, COBOL for MVS and VM, COBOL for OS/390 and VM, Enterprise COBOL for z/OS and OS/390, and Enterprise COBOL for z/OS.

OS/VS COBOL Reporting

MRS makes a distinction between data bytes and procedure bytes in the detailed report. Data bytes include the storage size in the CSECT that is directly generated by the COBOL CSECTs Data Division plus the INIT1 code. (Data bytes are figured from the beginning of the program to the TGT.) Procedure bytes include the storage size of the remainder of the CSECT. The sections of the CSECT that make up the procedure bytes are as follows:

- TGT
- PGT
- Report Writer Routines
- COBOL Procedure Division
- Q Routines
- INIT2
- INIT3

COBOL II and Above

When reporting for COBOL II and above, MRS distinguishes between data and procedure bytes. The data bytes include only the areas in the program generated at compile time, and do not include data areas acquired dynamically at execution time for a program compiled with the RENT option. The sections of the CSECT that make up the procedure bytes are as follows:

- Signature Code
- INITD Code (for RENT programs)
- Q Routines
- Procedure Division

Options Summary Report

The Options Summary produces a report detailing your installations default options.

- [JCL \(see page 313\)](#)
- [Report \(see page 313\)](#)

JCL

A procedure called CAIOPTNS is provided to invoke the Options Summary program. To obtain an options summary for your data center, specify the following:

```
// EXEC   CAI0PTNS
```

The initialization summary program, CAOUOPTS, uses the following DD statements:

DD Statement	Description
STEPLIB	Defines the library containing CA SymDump Batch Options Summary program and options modules.
SYSPRINT	Defines the report file.

Report

A sample of the Options Summary is as follows:

```
*
* EXECUTION TIME DEFAULTS:
*
* CEEDUMP      = OFF
* DUPLIM       = 5
* EXPDAYS      = 0
*              .
*              .
*              .
* SYSOUTD      =
* UNIT         = SYSDA
```

```

*
* CA0UXMOD   = CAO
* CA0UXMOD   = CEE
* CA0UXMOD   = IGZ
* CA0UXMOD   = ILB0
* CA0UXMOD   = IN25
*
* CA0USYM    = CAI.PROTSYM
*
*****
SUPPORT IS ACTIVE
SQA ANCHOR LOADED AT 82A3D2F0

REPORTING OPTIONS:

OPTION          STATUS          FIXED

ACB             = OFF
ACTONLY         = OFF
AMB             = OFF
.               .
.               .
WORKSTOR        = ON
WTOXCL          = ON

LOGGING OPTIONS:

OPTION          STATUS          FIXED

LOGTSO          = ON
LOGNET          = OFF
LOGROS          = ON
LOGUNI          = OFF
LOGWTO          = ON
LOGWTL          = OFF
SMF             = OFF

CONTROL OPTIONS:

MCHAR VALUE    = ?
PCHAR VALUE    = *
RCHAR VALUE    = :
SYSOUT VALUE   = *
MAXBLL         = 512

DEFAULT DATA SET NAMES:

NTMGCCTL       CAI.NTMGCCTL
NTMPI          CAI.NTMPI
NTMSC          CAI.NTMSC
CAIOPTS FILE SUFFIX      CAIOPTS

```

CSL Summary Report

The CSL Summary produces a report of all members in the Condensed Source Listing library. The user can select to report on a single CSL or on all of the CSLs defined as installation defaults.



Note: CA SymDump Batch no longer writes symbolic information using the CSL format. This utility provides summary information for CSL libraries that were populated by earlier versions of the product. These CSL libraries may still be used as input to the report formatter. However, as programs are recompiled, their updated symbolic information can only be written using the PROTSYM format.

- JCL (see page 315)
- Report (see page 315)

JCL

A procedure called CAICSLD is provided to invoke the Condensed Source Listing Summary program. To obtain a CSL Summary for your data center, specify the following:

```
// EXEC    CAICSLD
```

The CSL summary program, CAOUOPTS, uses the following DD statements:

DD Statement	Description
STEPLIB	Defines the library containing CA SymDump Batch CSL Summary program.
SYSPRINT	Defines the report file.
CAISYM	(Optional) Defines the Condensed Source Listing library to be reported on.



Note: The ddname for the CSL is an installation option that can be modified. If it is modified at your installation, be sure to also change the ddname in procedure CAOUCLSD. If omitted, CAOUCLSD reports on all of the CSLs defined as installation defaults.

Report

This sample report shows the information and the format of the condensed source listing summary:

```

                                CONDENSED SOURCE LIBRARY
                                DATA SET NAME = USER002.CSL
PROGRAM ID      DATE COMPILE   TIME COMPILED GENLEVEL  RELEASE  COMPILER
-- [A]-----  --- [B]-----  [C]-----  [D]-----  [E]-----  [F]---
ABEND          05 May,2004     12:55:21    00040X220    2.2      COBOL II
CALLEDTH       24 Nov,2003     16:42:27    99060X210    2.1      COB 0S390
CALLIVP1       23 Jan,2004     12:05:01    9912RX220    2.2      COB 0S390
CAOUTC2K       16 Jan,2004     16:56:51    9912HH100    1.0      COB 0S390
CA02DB2        05 Apr,2004     14:37:22    0002RX220    2.2      COB 0S390
CA02DEMO       14 Jul,2003     16:25:47    0007W9800    1.0      COB 0S390
CA02DEMR       06 Nov,2003     17:27:03    97090X210    2.1      COB 0S390
CA02IDMS       04 Apr,2003     13:29:58    0002RX220    2.2      COB 0S390
CCDEM003       08 Nov,2002     10:53:01    99090X220    2.2      COB 0S390
COB0C7         05 May,2004     14:20:08    0004RX220    2.2      COB 0S390
DB2ADD         26 Apr,2004     09:35:19    0106A0X00    3.0      COB 0S390
DB2CAF         24 Mar,2004     16:20:46    00020X220    2.2      COB 0S390
DB2DEL         26 Apr,2004     09:35:28    0106A0X00    3.0      COB 0S390
DB2DEMO        26 Apr,2004     09:35:10    0106A0X00    3.0      COB 0S390
DB2LIST        08 Sep,2002     18:24:28    99090X220    2.2      COB 0S390
DB2LISTM       07 Sep,2002     14:01:07    99090X220    2.2      COB 0S390
DB2LIST2       08 Sep,2002     18:25:05    99090X220    2.2      COB 0S390
DB2MSG         07 Sep,2002     14:08:38    99090X220    2.2      COB 0S390
DB2PLANS       25 Apr,2003     10:41:17    00020X220    2.2      COB 0S390
DB2TBLS        07 Mar,2003     15:07:54    00020X220    2.2      COB 0S390
DB2UPD         25 Apr,2003     09:35:39    0106A0X00    3.0      COB 0S390
DFSSAM01       14 Jul,2003     10:27:44    00070X220    2.2      COBOL II
ENTRIES        05 Feb,2002     17:08:55    9812HH100    1.0      COB 0S390
IC114A         11 Jul,2003     12:59:13    00070X220    2.2      COB 0S390

```

MAINPGM	05 Feb,2002	17:08:47	9812HH100	1.0	COB 0S390
MAINPROG	30 Apr,2002	17:09:41	99030X210	2.1	COB 0S390
MAIN1	06 DEC,2002	17:37:45	99120M610	6.1	VS COBOL
MWP503	11 Feb,2004	16:16:51	99120X220	2.2	COB 0S390
MWP635	08 May,2003	09:35:31	0004RX220	2.2	COBOL II
M648884A	06 Jun,2003	17:50:29	0106A0X00	3.0	COBOL II
OPIMSDMO	11 Jul,2003	18:27:50	0007RX220	2.2	COBOL MVS
PSIMPLE	13 JAN,2002	15:46:02	97120X210	2.1	COBOL MVS
STOPRUN	03 May,2004	10:48:28	0004RX220	2.2	COBOL II
SUBPGM	06 Oct,2002	16:09:37	98130X220	2.2	COBOL II
SUBPROG	20 May,2002	13:13:03	95040X130	1.3	COBOL II
SUBPROGA	30 Apr,2002	17:09:36	99030X210	2.1	COB 0S390
SVTEST	28 Mar,2003	11:18:37	00020X220	2.2	COBOL II
TESTMRG	11 Feb,2003	14:57:15	99120X220	2.2	COB 0S390
XACD912	26 Jun,2003	11:37:38	9912HH100	1.0	COB 0S390
TOTAL NUMBER OF COBOL II PROGRAMS			=	8	[G]
TOTAL NUMBER OF VS COBOL PROGRAMS			=	1	
TOTAL NUMBER OF COBOL FOR MVS PROGRAMS			=	2	
TOTAL NUMBER OF COBOL FOR 0S/390 PROGRAMS			=	28	
TOTAL NUMBER OF PROGRAMS			=	39	

The Condensed Source Listing Summary report displays the following information:

- **[A]**
Lists the program IDs.
- **[B]**
Displays the date when the program was compiled.
- **[C]**
Displays the time when the program was compiled.
- **[D]**
Displays the genlevel of CA SymDump Batch when the CSL was created.
- **[E]**
Displays the release of CA SymDump Batch when the CSL was created.
- **[F]**
Displays the compiler used.
- **[G]**
Displays the totals for each CSL reported on.

CAIPRINT Repository Utility

The CAIPRINT repository utility provides a batch facility to list and maintain the contents of the central VSAM repository.

- [JCL \(see page 316\)](#)
- [Commands \(see page 317\)](#)

JCL

A procedure called CAIPRTU is provided to invoke the CAIPRINT Repository Maintenance Utility, CAOUPRTU. To display or maintain the contents of your repository, specify the following:

```
// EXEC CAIPRTU
//PRTU.SYSIN DD *
(commands go here)
/*
```

CAOUPRTU uses the following DD statements:

DD Statement	Description
STEPLIB	Defines the library containing CA SymDump Batch.
SYSPRINT	Defines the report file.
SYSUT1	Defines the work file used for some requests.

Commands

The syntax of any command is:

keyword <selection criteria>

- **keyword**

Lets you specify one of the command keywords listed in this section.

- **selection criteria**

Lets you specify the selection criteria comprised of an argument and optionally an operator and a value.

Example:

```
LIST DATE GT 2004/04/06
```

In this example, the LIST keyword is used to display a summary of reports that were created in the repository after April 6, 2004. The selection criteria is comprised of the DATE argument, the GT (for Greater Than) operator and the value 2004/04/06.

Command Keywords

Following are the supported command keywords:

- **COPY**

Lets you copy reports from a secondary repository (ODSN) to your primary repository (DSN). Requires that both DSN and ODSN be specified.

- **DELETE**

Lets you delete selected reports from a repository (DSN). The DSN must be specified.

- **DSN**

Lets you define the dsname for your primary CAIPRINT repository. The dsname must be specified prior to using any other commands.

- **ERRSUM**

Lets a summary of abend reports contained within a repository (DSN). The summary includes the program name, offset, and abend code, and a count indicating the number of reports found with the same information. The entries are displayed in alphabetical order, by program name. Requires that DSN be specified.

- **INIT**
Lets you initialize a newly defined repository (DSN). Requires that DSN be specified.
- **JOBSUM**
Lists a summary of abend reports contained within a repository (DSN), by job name. The summary includes the job name, program name, offset, and abend code, and a count indicating the number of reports found with the same information. The entries are displayed in alphabetical order by job name and program name. Requires that DSN be specified.
- **LIST**
Lists a summary of reports contained within a repository (DSN). Requires that DSN be specified.
- **LOCK**
Lets you lock one or more reports in a repository (DSN) to prevent deletion. Requires that DSN be specified.
- **ODSN**
Lets you specify the dsname for a secondary CAIPRINT repository. Must be specified prior to using the COPY command.
- **UNLOCK**
Lets you unlock one or more locked reports in a repository (DSN) to enable the deletion. Requires that DSN be specified.



Note: The dsname must be specified prior to using any other commands.

Arguments

Valid arguments for selection criteria include the following:

- **AGE**
Lets you select reports by the number of days since they were created. This argument requires an operator and a 1- to 3-digit value.
Example:
`DELETE AGE GT 5`
- **ALL**
Lets you select every report unconditionally, with no operator or value specified.
Example:
`UNLOCK ALL`
- **CODE**
Lets you select reports by the completion code of the step that produced the report. This argument requires an operator and a 4- to 7-character value.
Example:
`LIST CODE EQ S0C7`
(or) `LIST CODE EQ U1125`

```
(or) LIST CODE EQ RC0004
(or) LIST CODE EQ CEE3250
```

System abend codes must begin with S followed by three hexadecimal digits.

User abend codes must begin with U followed by four decimal digits. Return codes must begin with RC followed by four decimal digits. IBM message codes must be 1- to 7-characters.

▪ DATE

Lets you select reports by the date on which they were created. This argument requires an operator and a value in the format yyyy/mm/dd.

Example:

```
DELETE DATE LT 2004/04/06
```

▪ JOB

Lets you select reports by the name of the job that produced the report. This argument requires an operator and a 1- to 8-character value.

Example:

```
UNLOCK JOB NE OPTJOB1
```

▪ LOCK

Lets you select reports by the ID of the user that owns a lock on the report. This argument requires an operator and a 1- to 7-character value.

Example:

```
LIST LOCK EQ OPTUSER
```

▪ PGM

Lets you select reports by the name of the abending program that caused the report. This argument requires an operator and a 1- to 8-character value.

Example:

```
DELETE PGM EQ CA02DEM0
```

▪ STEP

Lets you select reports by the name of the step that produced the report. This argument requires an operator and a 1- to 8-character value.

Example:

```
LIST STEP NE GO
```



Note: If the abend report was generated in an IMS DC environment under LE, STEP represents the Task ID.

▪ SYSTEM

Lets you select reports by the 1- to 8-character name of the system on which the report was created.

Example:

```
COPY SYSTEM EQ PR0DZ1
```

- **USER**

Lets you select reports by the ID of the user who submitted the job that produced the report. This argument requires an operator and a 1- to 7-character value.

Example:

LOCK USER EQ OPTUSER

Operators

Valid operators include the following:

- **EQ**

Selects reports if the value of the argument is equal to the value specified.

- **GE**

Selects reports if the value of the argument is greater than or equal to the value specified.

- **GT**

Selects reports if the value of the argument is greater than the value specified.

- **LE**

Selects reports if the value of the argument is less than or equal to the value specified.

- **LT**

Selects reports if the value of the argument is less than the value specified.

- **NE**

Selects reports if the value of the argument is not equal to the value specified.

- **RANGE**

Selects reports if the value of the argument is between the two values specified.

Dynamic Symbolic Support

The dynamic symbolic support feature, when activated, dynamically retrieves the compiler or assembler listing of the program being printed or viewed from a listing data set managed by CA Endevor SCM and loads it into the designated PROTSYM file. For this feature to work, the load module library (JOBLIB or STEPLIB) where the load module is loaded and the listing data set containing the module listing must be under the control of CA Endevor SCM.

- [Dynamic Symbolic Support Return Codes \(see page 321\)](#)
- [C1DEFLT5 Consideration \(see page 321\)](#)

Follow these steps:

1. Specify the option NDVRDSS=ON in your installation options table CAOUFRX using the CAIOPTS DD statement, or in the SYM Option 5 panel.

2. Designate a PROTSYM file to receive the dynamic populated symbolic using the NDVRDSN option.
The designated PROTSYM file can be any existing PROTSYM file and is searched ahead of all other PROTSYM files specified.

When the dynamic symbolic support feature is activated, it loads the correct symbolic information whenever a matching symbolic version is not found.

You can reload the symbolic files associated with programs that do not contain a time stamp in the executable, such as non-LE-enabled Assembler programs, every time they are printed or viewed. You can activate this behavior using the additional option NDVRASM=ON. Default is not to refresh the symbolic. This option, when suppressed (NDVRASM=OFF), never refreshes the symbolic of a non-LE-enabled Assembler program.



Note: The dynamic symbolic support feature cannot differentiate between multiple listing outputs created by a single CA Endeavor SCM processor for the same element. Thus, dynamic symbolic support using listing outputs from multiple compiles or assemblies from a single CA Endeavor SCM GENERATE or MOVE action for the same element may produce unpredictable results.

Dynamic Symbolic Support Return Codes

Dynamic symbolic support provides an integrated service using API calls that deploy various proven components including CA Endeavor SCM, PROTSYM post processors, z/OS dynamic allocation, CSVQUERY, and binder services.

For specific information about the return codes, see message CAPU309E in [Messages and Codes \(https://docops.ca.com/display/CAITSD11/Messages+and+Codes\)](https://docops.ca.com/display/CAITSD11/Messages+and+Codes).

C1DEFLT5 Consideration

Following are the considerations for single and multiple C1DEFLT5:

- Single C1DEFLT5 -- In a CA Endeavor SCM single C1DEFLT5 environment, dynamic symbolic support invokes CA Endeavor SCM API directly, provided the C1DEFLT5 data set and CA Endeavor SCM AUTHLIB and CONLIB data sets are in the LNKLIST.
- Multiple C1DEFLT5 -- In a CA Endeavor SCM multiple C1DEFLT5 environment, dynamic symbolic support normally invokes CA Endeavor SCM API through a listing server dedicated to a unique C1DEFLT5. For more information, see [Installing \(https://docops.ca.com/display/CAITSD11/Installing\)](https://docops.ca.com/display/CAITSD11/Installing).

Troubleshooting

This section contains information about identifying and resolving problems.

Collect Diagnostic Data

- [Execution-Time Problems \(see page 322\)](#)
- [Formatting Problems \(see page 322\)](#)
- [Symbolic Postprocessors \(see page 323\)](#)

Execution-Time Problems

Gather the following information on the execution environment:

- In an Online Data Communication environment, the name of the package (if it is not an in-house developed product), the version/release and maintenance level, and a vendor contact, if available.
- Non-COBOL programs being invoked by the program. Assembler language subroutines may be of interest to us in resolving the problem. Have the source available.
- The CAIPRINT report, if applicable.

Formatting Problems

Gather the following documentation:

- Print the member listing from the PROTSYM or CSL file. See examples later in this section for more information. For some problems, you may also be required to unload one or more members from a PROTSYM file using the batch utility IN25UTIL. For more information about unloading members with IN25UTIL, see [Symbolic Support \(https://docops.ca.com/display/CAITSD11/Symbolic+Support\)](https://docops.ca.com/display/CAITSD11/Symbolic+Support).
- Use IDCAMS REPRO to create a sequential file of your central VSAM repository. (If the repository is very large or you only want to send the data for a single report, you can copy the report to a temporary repository and REPRO that file.) See [Copying Repository Data for Diagnostic Purposes \(see page 324\)](#).
- For clients that do not use the repository, CA Technical Support may require the unformatted data used to generate the report. To provide that data, you will need to add the following DD to your execution time JCL and rerun the job:

```
//CAIPRTWK DD DSN=data-set-name,DISP=(NEW,CATLG,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(1,1)),
//          DCB=(LRECL=133,BLKSIZE=13300,RECFM=FBA)
```

The data set can then be sent to CA as a BINARY file.

Symbolic Postprocessors

Review all output from the postprocessor for error messages. If any messages are found, locate the message in [Symbolic Support](https://docops.ca.com/display/CAITSD11/Symbolic+Support) (<https://docops.ca.com/display/CAITSD11/Symbolic+Support>), then review the reason and suggested action(s).

Gather the following documentation:

- All output from the postprocessor
- The listing file used as input to the postprocessor

For all problems, gather:

- The full IBM system SYSUDUMP if an abend has occurred. Although there are other products that, like CA SymDump Batch, produce very neat, easy-to-read, condensed dumps appropriate for debugging user programs, they generally do not contain enough information for our purposes.
- All JCL, JOBLOG information, and messages for the job that ended in error.
- You may occasionally encounter a problem involving a COBOL program purchased from another vendor that is considered proprietary. While you are usually allowed to send the source listing on paper, you may not be permitted to send it on [tape](#). Check. Arrangements can usually be made, including the signing of nondisclosure agreements by CA.

Interpret Diagnostic Data

When you have collected the specified diagnostic data, write down your answers to the following questions:

1. What was the sequence of events prior to the error condition?
2. What circumstances existed when the problem occurred and what action did you take?
3. Has this situation occurred before? What was different then?
4. Did the problem occur after a particular PTF was applied or after a new version of the software was installed?
5. Have you recently installed a new version of the operating system or Language Environment (LE)?
6. Has the hardware configuration (tape drives, disk drives, and so forth) changed?

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem.

Print a Symbolic File Member for Diagnostic Purposes

There are two ways to print a symbolic file member or symbolic file directory. Either method is adequate for diagnostic purposes.

- [Use the Viewer \(see page 324\)](#)
- [Use the Batch Utilities \(see page 324\)](#)

Use the Viewer

Follow these steps:

1. Start the CA SymDump Batch Repository Viewer the way you normally do.
2. Enter the SYM primary command.
The Symbolic Utilities menu is displayed.
3. Select Option 1 (List) and type the name of your PROTSYM or CSL file in the field provided.
4. Press Enter to list the contents of your symbolic file
5. Use the PRINT primary command to print a copy of the display to a printer, file, or other desired destination.
6. After the directory is printed, type the **p** line command next to any member to print that member.

Use the Batch Utilities

Sample JCL is provided for three utility programs that can be used to display the contents of your symbolic files:

- CAOUCLSD -- Use this program to generate a member list for your CSL library.
- CAORMAIN -- Use the PRINTCSL function of CAORMAIN to print a member from your CSL library.
- IN25UTIL -- Use the REPORT function of IN25UTIL to generate a list of symbolic members in your PROTSYM and use the PRINT function to print a symbolic listing for any member.

Copy Repository Data for Diagnostic Purposes

CA Technical Support may require the data for one or more reports in your CAIPRINT repository for diagnostic purposes. To provide that data, you must create a sequential data set from your VSAM repository.

- [Create a Temporary VSAM Repository \(see page 325\)](#)
- [Copy the Report to a Temporary VSAM Repository \(see page 325\)](#)
- [Create a Sequential Data Set from a VSAM Repository \(see page 325\)](#)

If your CAIPRINT repository is very large, or you only want to send the data for selected reports, you can use the following steps:

1. Create a temporary VSAM repository.
2. Copy the required reports to that repository.
3. Create a sequential data set from the temporary repository.
Otherwise, if you want to send the data for all reports, you can skip directly to Step 3. Create a Sequential Data Set from a VSAM Repository.

Create a Temporary VSAM Repository

Member CARXREPO in the CAI.CAVHJCL library contains JCL to create and initialize a CAIPRINT repository.

Follow these steps:

1. Make a copy of the JCL member and modify the appropriate fields, such as the data set name, space, and volume. (Ensure to change the data set name in both the CLUSTER and DATA definitions.)
2. Submit the JCL to create your temporary repository.

Copy the Report to a Temporary VSAM Repository

Use the COPY command of the CAIPRINT Repository Utility, CAOUPRTU, to copy one or more reports from your central VSAM repository to the temporary repository you just created.

For example, the following JCL copies the report for job TESTJOB from the central VSAM repository (CAI.PRTLIB) to a temporary VSAM repository (TEMP.PRTLIB).

```
// EXEC CAIPRTU
//PRTU.SYSIN DD *
   ODSN  CAI.PRTLIB
   DSN   TEMP.PRTLIB
   COPY  JOB EQ TESTJOB
/*
```



Note: If there is more than one report in the central VSAM repository with a job name of TESTJOB, the example above copies all of the reports by that name.

Create a Sequential Data Set from a VSAM Repository

Use IDCAMS REPRO to create a sequential file from your VSAM repository. For example, the following JCL creates a sequential file (TEMP.PRTLIB.BACKUP) from a temporary VSAM repository (TEMP.PRTLIB).

```
//REPRO      EXEC PGM=IDCAMS
//PRTLIB     DD  DSN=TEMP.PRTLIB,DISP=SHR
//BACKUP     DD  DSN=TEMP.PRTLIB.BACKUP,DISP=(NEW,CATLG),
//           UNIT=SYSDA,VOL=SER=vvvvvv,SPACE=(CYL,(3,3),RLSE),
```

```
//          DCB=(RECFM=FB,LRECL=4085,BLKSIZE=4085)
//SYSPRINT DD  SYSOUT=*
//SYSIN      DD  *
  REPRO INFILE(PRTLIB) OFILE(BACKUP)
/*
```