

IBM Fault Analyzer for z/OS



# User's Guide and Reference

*Version 13 Release 1*

**Note**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 583.

#### **Edition notice**

This edition published 1 December 2015 (PI51864) applies to Version 13 Release 1 Modification Level 0 of IBM Fault Analyzer for z/OS (5655-Q11) and to any subsequent releases and modifications until otherwise indicated in new editions.

IBM® ships regular up-to-date copies of this User's Guide and Reference with the Fault Analyzer PTFs in Acrobat Reader format. This data set can be downloaded from MVS to your PC in BINARY mode from IDL.SIDIDOC1(IDIUGPDF) for viewing and printing.

For the Japanese feature of Fault Analyzer, the most recently translated Japanese edition is available in IDL.SIDIDJPN(IDIUGPDF).

For the Korean feature of Fault Analyzer, the most recently translated Korean edition is available in IDL.SIDIDKOR(IDIUGPDF).

These three data sets available on the mainframe MVS system on which Fault Analyzer is installed.

This publication is available on the web at: <http://www.ibm.com/software/awdtools/faultanalyzer>

**© Copyright IBM Corporation 2000, 2015.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this book . . . . .</b>	<b>xii</b>
Who should use this book . . . . .	xii
Organization of this book . . . . .	xii
How to read the syntax diagrams . . . . .	xii
<b>How to send your comments to IBM . . . . .</b>	<b>xv</b>
If you have a technical problem . . . . .	xv
<b>Summary of changes . . . . .</b>	<b>xvii</b>
SC19-4116-10: APAR PI51864 Edition (December 2015) . . . . .	xvii
SC19-4116-09: APAR PI48923 Edition (November 2015) . . . . .	xvii
SC19-4116-08: APAR PI40802 Edition (July 2015) . . . . .	xviii
SC19-4116-07: APAR PI35460 Edition (April 2015) . . . . .	xviii
SC19-4116-06: APAR PI28247 Edition (November 2014) . . . . .	xix
SC19-4116-05: APAR PI19883 Edition (August 2014) . . . . .	xix
SC19-4116-04: APAR PI17012 Edition (June 2014) . . . . .	xx
SC19-4116-03: APAR PI11055 Edition (April 2014) . . . . .	xx
SC19-4116-02: APAR PI08456 Edition (January 2014) . . . . .	xxi
SC19-4116-01: APAR PM99413 Edition (November 2013) . . . . .	xxi
SC19-4116-00: October 2013 . . . . .	xxi
<b>Part 1. Using Fault Analyzer . . . . .</b>	<b>1</b>
<b>Chapter 1. Introduction . . . . .</b>	<b>3</b>
The analysis engine . . . . .	3
The analysis process . . . . .	3
Real-time abend analysis . . . . .	3
Real-time SNAP analysis . . . . .	5
Fault reanalysis . . . . .	6
Fault history files . . . . .	8
Special members in the history file data set . . . . .	9
Supported application environments . . . . .	9
Binder-related dependencies . . . . .	11
Setting up existing programs for fault analysis . . . . .	12
Additional region size required . . . . .	12
Compiler listing or side file selection criteria . . . . .	12
Special processing of Language Environment CEEWUCHA abends . . . . .	13
WTO routing and descriptor codes used by Fault Analyzer . . . . .	13
CICS Storage Accounting Area (SAA) overlay assistance . . . . .	13
<b>Chapter 2. Real-time analysis . . . . .</b>	<b>15</b>
Dump suppression . . . . .	15
Fault Analyzer and CICS global user exits . . . . .	16
Fault history file selection . . . . .	16
Controlling the real-time analysis with options . . . . .	17
Pointing to listings with JCL DD statements . . . . .	18
The real-time analysis report . . . . .	18
Combining Fault Analyzer real-time reports . . . . .	19
Controlling the SYSOUT class of real-time reports . . . . .	19
Suppressing real-time reports . . . . .	19
The SYSLOG summary . . . . .	20
Using the program SNAP interface (IDISNAP) . . . . .	20
IDISNAP invocation . . . . .	21
Invoking Fault Analyzer from Java catch block . . . . .	25
Invoking Fault Analyzer from Java dump events . . . . .	26
Dump registration processing . . . . .	27
Real-time exclusion processing . . . . .	27
Duplicate fault processing . . . . .	30
Recovery fault recording . . . . .	30
RFR dump titles . . . . .	32
Using SLIP,COMP=0C4 with Fault Analyzer . . . . .	32
<b>Chapter 3. The Fault Analyzer ISPF interface . . . . .</b>	<b>33</b>
Invoking the interface . . . . .	34
ISPF split screen support . . . . .	34
The Fault Entry List display . . . . .	34
Using views . . . . .	37
Changing the history file or view displayed . . . . .	37
Fault entry list column configuration . . . . .	43
Sorting and matching fault entries . . . . .	47
Additional ways to match and select faults . . . . .	49
Applying an action to a particular fault . . . . .	52
History file properties . . . . .	53
New history file allocation . . . . .	55
Change fault history file settings . . . . .	57
Resetting history file access information . . . . .	58
Refreshing fault entry information . . . . .	59
Fault entry expiration control . . . . .	61
Action-bar pull-down menus . . . . .	61
Commands . . . . .	64
CE . . . . .	64
COLS . . . . .	64
COPY . . . . .	65
DISASM . . . . .	65
DSECT . . . . .	66
DUPS . . . . .	66
EXEC . . . . .	66
FIND . . . . .	67
INFO . . . . .	69
LOOKUP . . . . .	69
MATCH . . . . .	70
NEXT . . . . .	70
NOTELIST . . . . .	71
PREV . . . . .	71
QUIT . . . . .	71
REFRESH . . . . .	72
RESET . . . . .	72
RPTFIND . . . . .	72
RUNCHAIN . . . . .	73
SHOW . . . . .	73
SHOWFREE . . . . .	73

STCK . . . . .	74	Primary option: Fault Analyzer Options . . . . .	139
VIEWS . . . . .	74	Displaying associated storage areas . . . . .	140
Viewing a saved report . . . . .	74	Hiding the hex-value column . . . . .	141
Adding or removing blank lines . . . . .	76	Collapsing level 88 items . . . . .	142
Adding or removing help text . . . . .	76	Showing all COBOL base locators . . . . .	144
Setting preferred formatting width . . . . .	76	Expanding messages and abend codes . . . . .	144
Displaying user-selected message or abend code explanations . . . . .	77	Displaying source code . . . . .	145
Copying interactive displays to a file . . . . .	79	Displaying storage locations . . . . .	147
Displaying Fault Analyzer copyright and general usage information . . . . .	80	Displaying data areas . . . . .	149
Deleting history file entries . . . . .	80	Creating and managing user notes . . . . .	150
Changing deletion options through the Options menu . . . . .	81	Mapping storage areas using DSECT information . . . . .	154
Deleting when the confirmation display is shown . . . . .	81	IDIDSECT concatenation . . . . .	156
Deleting when the confirmation display is not shown . . . . .	82	Indexing your DSECT data sets (\$DINDEX member) . . . . .	157
Bulk deletions . . . . .	82	DSECT indexing utility (IDIPDSCU) . . . . .	157
Viewing fault entry information . . . . .	82	Displaying chained data areas . . . . .	157
Viewing the fault entry duplicate history . . . . .	87	Disassembling object code . . . . .	160
Copying history file entries . . . . .	90	Converting STORE CLOCK values . . . . .	162
Moving history file entries . . . . .	91	User-specific report formatting . . . . .	164
Transmitting history file entries . . . . .	91	Prompting for compiler listing or side file . . . . .	166
Security considerations . . . . .	92	Controlling prompting . . . . .	171
Specifying 64-bit addresses . . . . .	93	Data sets used for interactive reanalysis . . . . .	171
<b>Chapter 4. Performing batch reanalysis</b>	<b>95</b>	Refresh processing . . . . .	171
Batch reanalysis options . . . . .	95	COBOL Explorer . . . . .	172
Initiating batch reanalysis . . . . .	100	COBOL Explorer example . . . . .	173
Data sets used for batch reanalysis . . . . .	100	Deferred Breakpoints Feature . . . . .	176
Creating your own batch reanalysis job . . . . .	101		
<b>Chapter 5. Performing interactive reanalysis</b>	<b>103</b>		
Interactive reanalysis options . . . . .	103		
Initiating interactive reanalysis . . . . .	108		
Status pop-up display . . . . .	108		
General information about the interactive report . . . . .	109		
Exit from the interactive report . . . . .	111		
Primary option: Synopsis . . . . .	112		
Primary option: Event Summary . . . . .	112		
Detailed Event Information . . . . .	114		
Primary option: Open Files . . . . .	117		
Primary option: CICS Information . . . . .	119		
CICS Transaction Storage Summary . . . . .	120		
Last CICS 3270 Screen Buffer . . . . .	120		
Last CICS 3270 Screen Buffer Hex . . . . .	121		
Summarized CICS Trace . . . . .	122		
CICS Trace Formatting . . . . .	123		
CICS Levels, Commareas, and Channels . . . . .	126		
Primary option: Messages . . . . .	128		
Primary option: DB2 Information . . . . .	129		
Primary option: IMS Information . . . . .	131		
Primary option: Storage Areas . . . . .	136		
Primary option: Java Information . . . . .	137		
Primary option: Language Environment Heap Analysis . . . . .	137		
Primary option: MTRACE Records . . . . .	138		
Primary option: Abend Job Information . . . . .	139		
Primary option: User Notes . . . . .	139		
<b>Chapter 6. Performing CICS system abend dump analysis</b>	<b>177</b>		
Setting options for CICS system abend analysis . . . . .	177		
User exits . . . . .	177		
Selecting a CICS dump data set . . . . .	177		
Selecting an address space to analyze . . . . .	179		
Displaying the CICS system abend interactive report . . . . .	180		
Fastpath navigation . . . . .	181		
Option 1: Synopsis . . . . .	181		
Option 2: Abend Job Information . . . . .	181		
Option 3: CICS System Information . . . . .	182		
Option 4: Options in Effect . . . . .	184		
Creating a history file entry . . . . .	185		
<b>Chapter 7. Formatting a CICS auxiliary trace data set</b>	<b>187</b>		
Selecting a CICS auxiliary trace data set . . . . .	187		
Specifying CICS Trace Selection Parameters . . . . .	188		
<b>Chapter 8. Performing Java analysis</b>	<b>189</b>		
Setting options for Java analysis . . . . .	189		
Selecting a Java dump data set . . . . .	189		
Java fault entry reanalysis . . . . .	190		
Displaying the Java information in the interactive report . . . . .	191		
<b>Chapter 9. The Fault Analyzer report</b>	<b>197</b>		
General report information . . . . .	197		
Most significant abend code . . . . .	197		
Open file record information . . . . .	197		
COBOL suppressed copybooks . . . . .	199		
Main report sections . . . . .	199		

The prolog section . . . . .	200
The synopsis section . . . . .	200
The summary section . . . . .	200
The event details section . . . . .	202
The system-wide information section . . . . .	203
The abend job information section . . . . .	204
The options section . . . . .	204
The epilog section . . . . .	204
Sample reports . . . . .	204
<b>Chapter 10. Using non-ISPF interfaces to access Fault Analyzer history files . . . . .</b>	<b>205</b>
Using the Fault Analyzer plug-in for Eclipse . . . . .	205
Using the Fault Analyzer client for IBM Rational Developer for System z . . . . .	205
Performing interactive reanalysis under CICS . . . . .	206
Using the Fault Analyzer web interface. . . . .	206
<b>Part 2. Fault Analyzer installation and administration . . . . .</b>	<b>209</b>
<b>Chapter 11. Migrating from an earlier version of Fault Analyzer . . . . .</b>	<b>211</b>
Migrating from V12.1 to V13.1. . . . .	211
Migrating from V11.1 to V12.1. . . . .	212
Migrating from V10.1 to V11.1. . . . .	212
Migrating from V9.1 to V10.1 . . . . .	213
Migrating from V8.1 to V9.1 . . . . .	213
Migrating from V7.1 to V8.1 . . . . .	213
Migrating from V6.1 to V7.1 . . . . .	214
LPA module compatibility . . . . .	216
Sharing of history files across a sysplex with mixed levels of Fault Analyzer . . . . .	216
<b>Chapter 12. Preparing to customize Fault Analyzer . . . . .</b>	<b>217</b>
Checklist for installing and customizing Fault Analyzer . . . . .	217
Library names after you finish installing . . . . .	221
Storage recommendations . . . . .	222
Exits for invoking Fault Analyzer. . . . .	223
Invocation for non-CICS transaction abends . . . . .	223
Invocation for CICS transaction abends. . . . .	227
SVC dump registration . . . . .	227
Language Environment options required for invocation of Fault Analyzer . . . . .	228
LE options required for non-CICS abends . . . . .	228
LE options required for CICS abends . . . . .	228
Running Fault Analyzer with similar third-party products . . . . .	228
MVS dump data set size. . . . .	229
Application-handled error conditions . . . . .	229
Setting Fault Analyzer SLIP traps. . . . .	229
<b>Chapter 13. Customizing the operating environment for Fault Analyzer. . . . .</b>	<b>231</b>
Making Fault Analyzer modules available . . . . .	231
Defining program control access to Fault Analyzer programs. . . . .	232
Restricting change of history file settings . . . . .	233
Setting up the message and abend code explanation repository . . . . .	233
Managing recovery fault recording data set access . . . . .	234
SDUMP recovery fault recording data sets. . . . .	235
TDUMP recovery fault recording data sets . . . . .	236
RFR TDUMP XFACILIT example. . . . .	237
Registering Fault Analyzer in the IFAPRDxx parmlib member . . . . .	238
Generic product enablement . . . . .	238
<b>Chapter 14. Using the Fault Analyzer IDIS subsystem . . . . .</b>	<b>239</b>
Sysplex-wide subsystem inter-communication . . . . .	240
Caching of history file \$\$INDEX data . . . . .	240
Starting the IDIS subsystem . . . . .	241
IDIS subsystem storage requirements . . . . .	243
IDIS subsystem requirements for DB2 . . . . .	243
IDIS subsystem requirements for Java . . . . .	244
Stopping the IDIS subsystem . . . . .	244
<b>Chapter 15. Modifying your ISPF environment . . . . .</b>	<b>247</b>
Allocating ISPF data sets . . . . .	247
Making the Fault Analyzer IDISCMDS command table available . . . . .	247
Updating the ISPF selection panel . . . . .	248
Invoking Fault Analyzer using an ISPF 3.4 line command (FA) . . . . .	248
Invoking Fault Analyzer from SDSF (SFA command) . . . . .	249
Invoking the LOOKUP command using cursor selection (LOOKC command) . . . . .	250
Providing ISPF interface defaults for new users . . . . .	250
Providing installation-specific batch reanalysis JCL control statements . . . . .	250
Increasing the display area for Fault Analyzer reports . . . . .	251
<b>Chapter 16. Customize Fault Analyzer by using USERMODs . . . . .</b>	<b>253</b>
Enabling Fault Analyzer to be invoked . . . . .	253
Installing the MVS change options/suppress dump exit IDIXDCAP . . . . .	253
Enabling the Language Environment abnormal termination exit IDIXCEE . . . . .	254
Working with applications that use a non-Language Environment run time . . . . .	255
Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++IDISPLI/++IDISPLIA) . . . . .	255
Always invoking Fault Analyzer from PL/I PLIDUMP (++IDISPDM) . . . . .	255
Eliminating the need for a dump DD statement (++IDITABD) . . . . .	255
Obtaining load modules from CA-Panexec . . . . .	256
Using a non-standard LE parameter list separator character (++IDIOPT1) . . . . .	256

Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (++IDIOPT2) . . . . .	256
--	-----

## **Chapter 17. Customize Fault Analyzer by using an IDIOPTLM configuration-options module . . . . .**

Identifying the LE runtime library . . . . .	257
Specifying an alternative parmlib data set for IDICNFxx . . . . .	258
Changing the default recovery fault recording IEATDUMP data set name . . . . .	258
Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit .	258

## **Chapter 18. Setting up history files 261**

Determining what size history files to allocate . . . . .	261
Allocating a PDS or PDSE for a history file . . . . .	261
AUTO-managed PDSE history files . . . . .	262
Providing the name of a history file to Fault Analyzer . . . . .	262
Setting up views . . . . .	263
Specifying a default column layout . . . . .	264
Specifying an initial fault entry selection criteria .	264
View considerations if not using XFACILIT resource class . . . . .	265
Managing history files across MVS systems without shared DASD . . . . .	266
Automated implementation . . . . .	266
On demand implementation . . . . .	270
Sharing of history files across a sysplex . . . . .	272
Managing history file fault entry access . . . . .	273
Using the XFACILIT resource class for history file fault entries . . . . .	274

## **Chapter 19. Setting and changing default options for the site . . . . .**

Parmlib member IDICNFxx . . . . .	279
Controlling which jobs are analyzed with Exclude processing . . . . .	281
Fast Exclude options processing . . . . .	282
Controlling report detail . . . . .	282
Limiting the size of minidumps . . . . .	282
Pointing to listings and REXX exec libraries . . . . .	283
Suppressing noncritical SYSLOG messages . . . . .	283
Customizing the Fault Entry List display . . . . .	283
Specifying the cultural environment . . . . .	283

## **Chapter 20. Providing compiler listings or Fault Analyzer side files . . . . .**

COBOL Report Writer Precompiler . . . . .	286
Required compiler options for IDILANGX . . . . .	286
TEST option considerations . . . . .	288
DEBUG option considerations . . . . .	288
Naming compiler listings or side files . . . . .	289
Naming CSECTs for Fault Analyzer . . . . .	289
Locating compiler listings or side files . . . . .	290
IDITRACE information . . . . .	292
Using the IDIXSFOR compiler listing/side file search and override exit . . . . .	292

IDIXSFOR invocation . . . . .	293
Compiler listings and side file attributes . . . . .	296
Using the IDIRLOAD DDname for CSECT mapping . . . . .	297
IDIDATST side file availability test utility . . . . .	298
ISPF-packed compiler listings . . . . .	299

## **Chapter 21. Customizing the CICS environment . . . . .**

Configuring Language Environment for CICS to invoke Fault Analyzer . . . . .	302
Defining required programs to CICS . . . . .	302
Adding the required programs to the startup PLT .	303
Adding the required programs to the shutdown PLT . . . . .	303
Enabling dynamic control of analysis of CICS transaction abends . . . . .	303
Using CFA to FORCEPURGE the currently analyzed task . . . . .	303
SVC dump screening . . . . .	304
Sample definition job . . . . .	304
Controlling CICS transaction abend analysis . . . . .	305
Using CFA from a CICS terminal . . . . .	305
Using CFA from an MVS console . . . . .	308
Ensuring transaction abend analysis is not suppressed by DUMP(NO) . . . . .	310
CICS NoDup(CICSFAST) override assembler exit (IDINDFUE) . . . . .	310
Invocation . . . . .	310
CICS trace considerations . . . . .	312
Preventing LE from causing the CICS trace to wrap . . . . .	312
Specifying CICS options through the IDIOPTS DDname . . . . .	312
Language Environment abend considerations . . . . .	313
Capture of abends running on CICS user key open TCBs (L9 TCBs) . . . . .	313
Installing the MVS post-dump exit IDIXTSEL . . . . .	313
Storage requirements . . . . .	313
Maximizing CICS transaction abend analysis performance . . . . .	314
Implementing an XEIIN global user exit . . . . .	314

## **Chapter 22. Customizing the DB2 environment . . . . .**

Binding DB2 . . . . .	317
DB2 and Language Environment . . . . .	317
Improving Fault Analyzer DB2 performance . . . . .	317

## **Chapter 23. Customizing the IMS environment . . . . .**

IMS and Language Environment . . . . .	319
--	-----

## **Chapter 24. Customizing the Fault Analyzer Japanese feature . . . . .**

Allocating ISPF data sets . . . . .	321
Setting the national language . . . . .	321

<b>Chapter 25. Customizing the Fault Analyzer Korean feature</b>	<b>323</b>	<b>Chapter 29. Maintaining Fault Analyzer</b>	<b>353</b>
Allocating ISPF data sets	323		
Setting the national language	323		
<b>Chapter 26. Verifying the customization of Fault Analyzer</b>	<b>325</b>	<b>Chapter 30. Disabling Fault Analyzer</b>	<b>355</b>
Verifying the use of Fault Analyzer with assembler	325	Temporarily uninstalling Fault Analyzer	355
Verifying the use of Fault Analyzer with COBOL	326	Turning off Fault Analyzer using the IFAPRDxx parmlib member	355
Verifying the use of Fault Analyzer with PL/I	327	Turning off Fault Analyzer with a JCL switch (IDIOFF)	356
Verifying the use of Fault Analyzer with C	328	Turning off Fault Analyzer using an environment variable (_IDI_OFF)	357
Verifying the IDIXCEE Language Environment exit enablement	329		
Verifying the IDITABD USERMOD installation	329		
Verifying the customization of Fault Analyzer under CICS	329		
CICS IVP: 0C1 in program IDIXFA	330		
CICS IVP: EXEC CICS DUMP			
DUMPCODE(FAD1)	330		
CICS IVP: EXEC CICS ABEND ABCODE(FLT1)	330		
CICS IVP: EXEC CICS ABEND ABCODE(FLT2)	331		
Verifying the use of Fault Analyzer with DB2	331		
Using a C program	331	Tracing user exit parameter list values	363
Using a COBOL program	333	Tracing REXX EXECs	367
Verifying the use of Fault Analyzer through ISPF	335		
Verifying the recovery fault recording set-up	335		
<b>Chapter 27. Managing history files (IDIUTIL utility)</b>	<b>337</b>	<b>Chapter 31. Customizing Fault Analyzer by using user exits</b>	<b>359</b>
IDIUTIL control statements	338	Invocation parameters	362
FILES control statement	338	Global environment data area (ENV)	362
LISTHF control statement	338	User exit type specific data area	362
DELETE control statement	339	Supported exit programming languages	362
SETFAULTPREFIX control statement	340	Data area version checking	363
SETMAXFAULTENTRIES control statement	341	Diagnostic tracing	363
SETMINFAULTENTRIES control statement	342	Tracing user exit parameter list values	363
IMPORT control statement	342	Tracing REXX EXECs	367
EXITS control statement	343		
Examples	344	Descriptions of fault analysis user exit types	367
Example 1. Listing history file entries	344	Analysis Control user exit	368
Example 2. Deleting history file entries by date	344	Analysis Control user exit (MVS SVC Dump registration)	371
Example 3. Deleting history file entries by utilization	345	Compiler Listing Read user exit	372
Example 4. Changing history file fault prefix characters	345	Message and Abend Code Explanation user exit	377
Example 5. Creating a self-maintained history file	345	Formatting user exit	381
Example 6. Importing history file entries	346	End Processing user exit	393
IDIUTIL batch utility user exit samples	347	End Processing user exit (Fault entry refresh)	396
<b>Chapter 28. Providing application-specific explanations and descriptions</b>	<b>349</b>	Notification user exit	397
User-defined message explanations	349	Notification user exit (MVS SVC Dump registration)	402
User-defined abend code explanations	350	Descriptions of IDIUTIL batch utility user exit types	403
User-defined program descriptions	351	IDIUTIL Import user exit	403
		IDIUTIL Delete user exit	404
		IDIUTIL ListHF user exit	405
		User exit REXX commands	409
		Evaluate command	409
		IDIALLOC command	411
		IDIDDTEST command	415
		IDIDSECTdsn command	415
		IDIEventInfo command	416
		IDIFREE command	417
		IDIModQry command	418
		IDIRegisterFaultEntry command	418
		IDIWRITE command	420
		IDIWTO command	421
		List command	422
		Note command	423
		Formatting tags	425
		ADDR (address)	428
		AREA (area)	428
		DD (definition description)	429
		DATA (data)	429
		DL (definition list)	429
		DT (definition term)	430
		DUMP (EBCDIC dump)	431

DUMPA (ASCII dump) . . . . .	431
HP (highlighted phrase) . . . . .	432
L (line) . . . . .	432
LI (list item) . . . . .	433
NOTEL (note list) . . . . .	433
P (paragraph) . . . . .	433
TH (table heading) . . . . .	434
U (underline) . . . . .	434
UL (unordered list) . . . . .	435
The IDIXUFMT load module Formatting user exit	435
Entry specifications . . . . .	436
Return specifications . . . . .	436
Sample IDIXUFMT exits. . . . .	436
IDIXUFMT functions . . . . .	436
<b>Chapter 32. Installing non-ISPF interfaces to access Fault Analyzer history files . . . . .</b>	<b>443</b>
Installing the Fault Analyzer plug-in for Eclipse	443
Customize the IBM Problem Determination Tools for z/OS Common Component Common Server . . . . .	443
Download and install CICS Explorer . . . . .	444
Download Fault Analyzer plug-in . . . . .	444
Install Fault Analyzer plug-in into CICS Explorer . . . . .	444
Installing the Fault Analyzer client for IBM Rational Developer for System z . . . . .	444
Enabling interactive reanalysis under CICS . . . . .	445
Making the required CICS resource definitions . . . . .	445
Making the required CICS JCL changes. . . . .	446
Installing the Fault Analyzer web interface . . . . .	446
Running the sample job to copy server files into HFS . . . . .	446
Applying the required RACF (or equivalent) rules . . . . .	446
Customizing the server configuration . . . . .	447
Customizing the sample Liberty procedures . . . . .	447
Starting the Liberty profile . . . . .	448
<b>Part 3. Fault Analyzer reference information . . . . .</b>	<b>449</b>
<b>Chapter 33. Options . . . . .</b>	<b>451</b>
Where to specify options . . . . .	452
Parmlib member IDICNF00. . . . .	453
User-options module IDICNFUM. . . . .	453
The _IDI_OPTSFILE environment variable. . . . .	454
User options file IDIOPTS . . . . .	454
The JCL EXEC statement PARM field . . . . .	455
The _IDI_OPTS environment variable . . . . .	455
Option descriptions . . . . .	455
AdditionalIDIOffDD . . . . .	455
CICSDumpTableExclude. . . . .	456
CICSTraceMax . . . . .	456
DataSets . . . . .	457
DeferredReport. . . . .	463
Detail . . . . .	465
DumpDSN . . . . .	466
DumpRegistrationExits . . . . .	466
ErrorHandler . . . . .	468
Exclude/Include . . . . .	469
Exits . . . . .	473
FAISPFopts . . . . .	475
FaultID . . . . .	478
GenerateSavedReport. . . . .	478
HistCols . . . . .	479
Include . . . . .	479
InteractiveExitPromptSeconds . . . . .	479
Language. . . . .	479
Locale . . . . .	480
LoopProtection . . . . .	481
MaxMinidumpPages . . . . .	482
NoDup . . . . .	482
PDTCCopts . . . . .	491
PermitLangx. . . . .	491
PreferredFormattingWidth . . . . .	494
PrintInactiveCOBOL . . . . .	494
Quiet . . . . .	495
RDZClient . . . . .	495
RefreshExits . . . . .	496
RetainCICSDump . . . . .	497
RetainDump. . . . .	498
Source. . . . .	499
SpinIDIREPRT . . . . .	499
StoragePrintLimit . . . . .	499
StorageRange . . . . .	501
SystemWidePreferred. . . . .	502
UseIDISTime . . . . .	503
<b>Chapter 34. Data areas . . . . .</b>	<b>505</b>
Non-REXX user exit buffered data format . . . . .	505
Data area descriptions . . . . .	506
CTL - Analysis Control user exit parameter list . . . . .	506
ENV - Common exit environment information . . . . .	513
EPC - End Processing user exit parameter list . . . . .	521
LST - Compiler Listing Read user exit parameter list . . . . .	522
NFY - Notification user exit parameter list. . . . .	524
UFM - Formatting user exit parameter list. . . . .	525
UTL - IDIUTIL Batch Utility user exit parameter list . . . . .	530
XPL - Message and Abend Code Explanation user exit parameter list . . . . .	530
<b>Chapter 35. Return codes . . . . .</b>	<b>533</b>
Batch reanalysis (IDIDA) . . . . .	533
IDIUTIL batch utility . . . . .	533
IDILANGX . . . . .	533
<b>Chapter 36. Messages . . . . .</b>	<b>535</b>
Fault Analyzer messages . . . . .	535
IDILANGX messages . . . . .	558
<b>Part 4. Appendixes . . . . .</b>	<b>559</b>

<b>Appendix A. Support resources and problem solving information . . . . .</b>	<b>561</b>
Searching knowledge bases. . . . .	561
Searching the information center . . . . .	561
Searching product support documents . . . . .	561
Getting fixes. . . . .	563
Subscribing to support updates . . . . .	563
RSS feeds and social media subscriptions . . . . .	563
My Notifications . . . . .	563
Contacting IBM Support. . . . .	564
Define the problem and determine the severity of the problem . . . . .	565
Gather diagnostic information. . . . .	565
Submit the problem to IBM Support. . . . .	566
<b>Appendix B. Sample customized ISPF interface front-end . . . . .</b>	<b>569</b>
Fault History File or VIEW name. . . . .	571
MATCH on program name. . . . .	571
Installing the sample application . . . . .	571
How the sample works . . . . .	571
<b>Appendix C. Java API to download</b>	
<b>Fault Analyzer report . . . . .</b>	<b>573</b>
Specification. . . . .	573
Example . . . . .	573
<b>Appendix D. Technical details for screen size adjustments. . . . .</b>	<b>581</b>
<b>Notices . . . . .</b>	<b>583</b>
Programming interface information . . . . .	584
Trademarks . . . . .	585
<b>Glossary . . . . .</b>	<b>587</b>
<b>Index . . . . .</b>	<b>589</b>



---

## About this book

This book tells you how to install and use Fault Analyzer to analyze user application abends.

---

## Who should use this book

This book is intended for any programmer responsible for the development or maintenance of any application that has been developed under one of the supported languages. Fault Analyzer is suited to problem determination in the application development, testing, or production environment.

System programmers might also want to consult this book, to find out how to install Fault Analyzer.

---

## Organization of this book

Part 1, “Using Fault Analyzer,” on page 1 provides an overview of Fault Analyzer. It discusses how Fault Analyzer is invoked when a user application abends, and tells you about the different modes of operation of Fault Analyzer.

Part 2, “Fault Analyzer installation and administration,” on page 209 describes how to set up Fault Analyzer so that it can analyze abends. It also tells you about user exits that can be used to influence Fault Analyzer execution.

Part 3, “Fault Analyzer reference information,” on page 449 provides information about options, return codes, data areas, and messages that are issued by Fault Analyzer.

Appendix A, “Support resources and problem solving information,” on page 561 contains information about IBM web sites which can help you answer questions and solve problems.

---

## How to read the syntax diagrams

The syntactical structure of commands that are described in this document is shown by means of syntax diagrams.

Figure 1 on page xii shows a sample syntax diagram that includes the various notations that are used to indicate such things as whether:

- An item is a keyword or a variable.
- An item is required or optional.
- A choice is available.
- A default applies if you do not specify a value.
- You can repeat an item.

## How to read the syntax diagrams

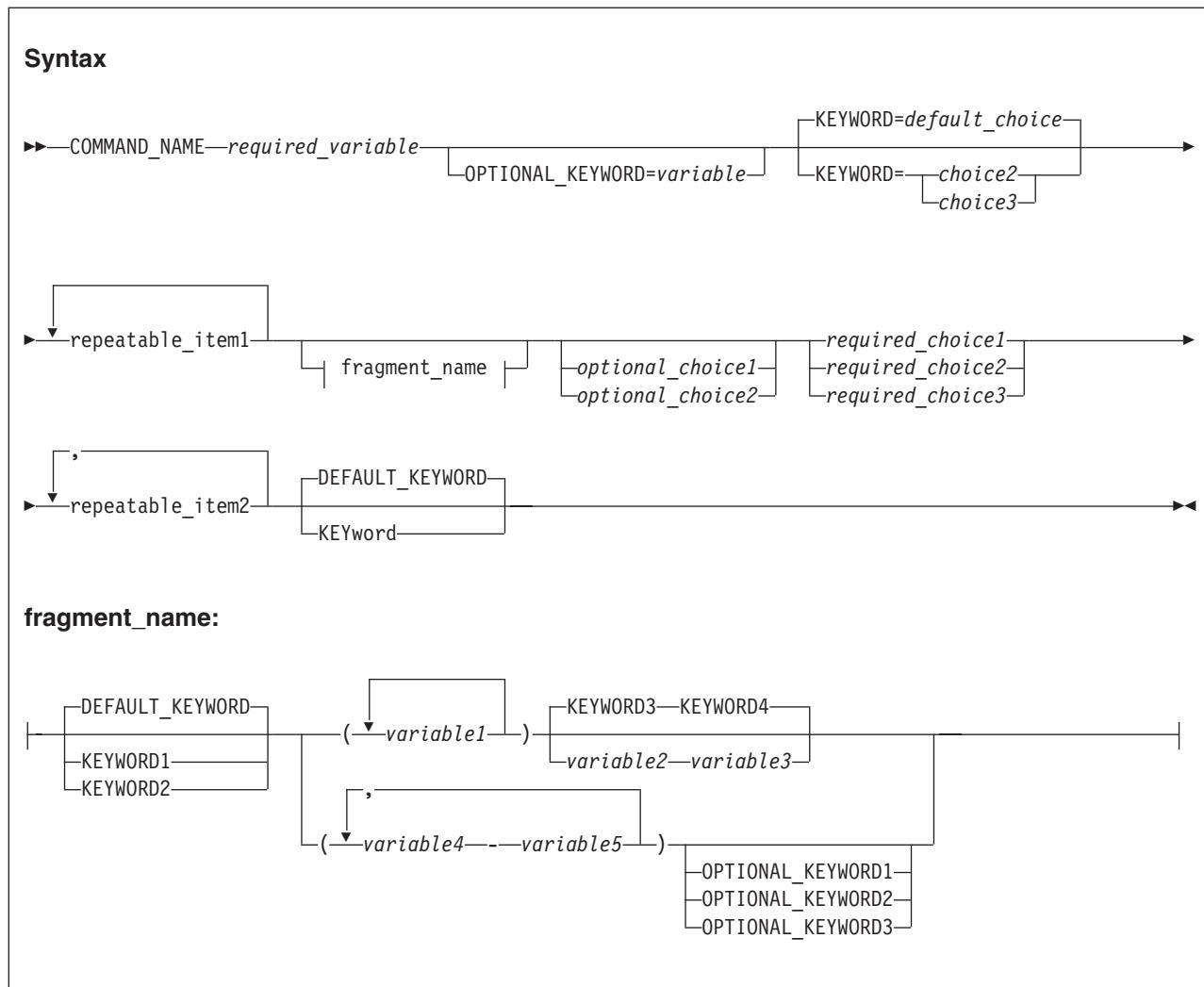


Figure 1. Sample syntax diagram

Here are some tips for reading and understanding syntax diagrams:

### Order of reading

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ►— symbol indicates the beginning of a statement.

The —→ symbol indicates that a statement is continued on the next line.

The ►— symbol indicates that a statement is continued from the previous line.

The —→► symbol indicates the end of a statement.

### Keywords

Keywords appear in uppercase letters.

►—COMMAND\_NAME—►►

## How to read the syntax diagrams

Sometimes you only need to type part of a keyword. The required part of the keyword appears in uppercase letters.



In this example, you could type "KW" or "KEYWORD".

The abbreviated or whole keyword you enter must be spelled exactly as shown.

### Variables

Variables appear in lowercase letters. They represent user-supplied names or values.



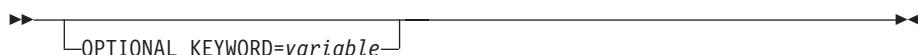
### Required

items Required items appear on the horizontal line (the main path).



### Optional

items Optional items appear below the main path.



### Choice of

items If you can choose from two or more items, they appear vertically, in a stack.

If you *must* choose one of the items, one item of the stack appears on the main path.



If choosing one of the items is optional, the entire stack appears below the main path.



If a default value applies when you do not choose any of the items, the default value appears above the main path.



## How to read the syntax diagrams

### Repeatable items

An arrow returning to the left above the main line indicates an item that can be repeated.

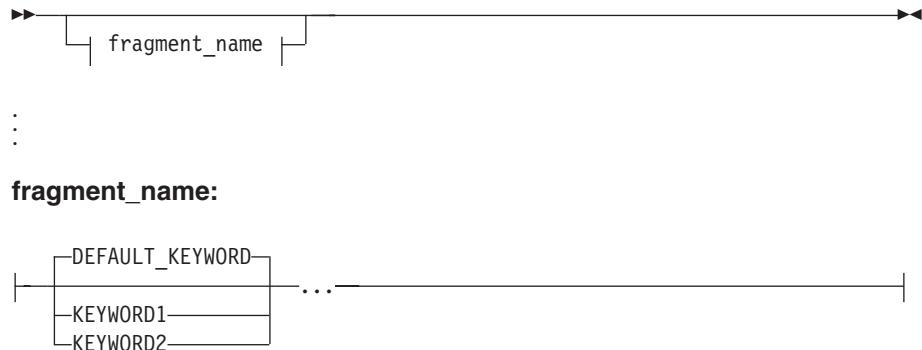


If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.



### Fragments

Where it makes the syntax diagram easier to read, a section or *fragment* of the syntax is sometimes shown separately.



---

## How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to [comments@us.ibm.com](mailto:comments@us.ibm.com)
2. Use the form on the web at <http://www.ibm.com/software/ad/rcf>
3. Mail the comments to the following address:

IBM Corporation  
DTX/E269  
555 Bailey Avenue  
San Jose, CA  
95141-1003  
U.S.A.

Include the following information:

- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:  
    IBM Fault Analyzer for z/OS User's Guide and Reference  
    SC19-4116-10
- The topic and page number that is related to your comment.
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

IBM or any other organizations only uses the personal information that you supply to contact you about the issues that you submit.

---

## If you have a technical problem

Do not use the feedback methods that are listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM support portal at <http://www.ibm.com/systems/z/support/>



---

## Summary of changes

The following describes the changes for this version.

---

### SC19-4116-10: APAR PI51864 Edition (December 2015)

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- The ability to change the way general purpose registers are displayed in the interactive reanalysis report has been provided. For details, see “Detailed Event Information” on page 114.
- Two new fields have been added to the ENV data area, ENV.ORIGINAL\_DATE and ENV.ORIGINAL\_TIME. For details, see “End Processing user exit” on page 393 and “ENV - Common exit environment information” on page 513.
- Information about Fault Analyzer support for z/OS V2R2 has been provided. For details, see “Supported application environments” on page 9.
- Information about the creation of LANGX side files using the IDILANGX utility has been moved to the *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

This edition of the book is provided in PDF format with the PTF for APAR PI51864 in data set IDI.SIDIDOC1(IDIUGPDF).

Changes to this edition of the book are identified by a vertical rule in the left-hand margin.

---

### SC19-4116-09: APAR PI48923 Edition (November 2015)

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Information about the user ID who last changed the Lock Flag setting of a fault entry has been provided. For details, see “Viewing fault entry information” on page 82.  
The user ID has also been made available to user exits via a new read-only ENV.LOCK\_USERID field. For details, see “ENV - Common exit environment information” on page 513.
- The installation of the Fault Analyzer MVS change options/suppress dump exit, IDIXDCAP, has changed for z/OS 2.2 and later versions of z/OS. For details, see “Installing the MVS change options/suppress dump exit IDIXDCAP” on page 253.
- Information about IPVLANGP, including a new Deferred Breakpoints Feature, has been moved to the *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

The same Deferred Breakpoints Feature has been provided with COBOL Explorer and a new interactive reanalysis option to specify the Debug Tool repository data set has been provided. For details, see “Interactive reanalysis options” on page 103..

- The ability to “Hold” history files in the list shown with the Last Accessed Fault History Files or Views display has been provided. For details, see “Changing the history file or view displayed” on page 37.

## **SC19-4116-09: APAR PI48923 Edition (October 2015)**

- Information about enabling and disabling Fault Analyzer using the IFAPRDxx parmlib member, when COBOL DevOps Tools Suite for z/OS V1.1 (5697-CDT) is installed, has been provided. For details, see “Registering Fault Analyzer in the IFAPRDxx parmlib member” on page 238 and “Turning off Fault Analyzer using the IFAPRDxx parmlib member” on page 355.

---

## **SC19-4116-08: APAR PI40802 Edition (July 2015)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Information about Fault Analyzer registration in the IFAPRDxx parmlib member for sites using generic product enablement has been added. For details, see “Registering Fault Analyzer in the IFAPRDxx parmlib member” on page 238.
- The sample exec provided to invoke Fault Analyzer from an ISPF 3.4 data set list, has been enhanced to include support for CICS auxiliary trace data sets. For details, see “Invoking Fault Analyzer using an ISPF 3.4 line command (FA)” on page 248.
- The RUNCHAIN command dialog has been 64-bit enabled. For details, see “Displaying chained data areas” on page 157.
- The DSECT command dialog has been 64-bit enabled. For details, see “Mapping storage areas using DSECT information” on page 154.
- The Formatting user exit Evaluate command address parameter has been 64-bit enabled. For details, see “Evaluate command” on page 409.
- The Formatting user exit List command address parameter has been 64-bit enabled. For details, see “List command” on page 422.
- The IDILANGP side file formatting utility can now be invoked from the Services action-bar pull-down menu of any ISPF interface display. For details, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.
- Two new ++USERMODs have been provided:
  - “Using a non-standard LE parameter list separator character (++IDIOPT1)” on page 256.
  - “Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (++IDIOPT2)” on page 256.

This edition of the book is provided in PDF format with the PTF for APAR PI40802 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-07: APAR PI35460 Edition (April 2015)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Additional sample Formatting user exits have been provided. For details, see “Additional sample Formatting user exits” on page 393.
- The ENV.INVOCATION\_EXIT data area field has been changed to include identification of CICS XDUREQ exit invocations, which are the result of EXEC CICS DUMP calls. For details, see “ENV - Common exit environment information” on page 513.
- The ability to select either a detailed or a summary format of the IMS Information display has been provided. For details, see “Primary option: IMS Information” on page 131.

- An additional suboption, UserAbendPgm(INclude | EXclude), has been provided for the ErrorHandler option. For details, see “ErrorHandler” on page 468.
- Enhancements have been made to the display of data areas within the interactive reanalysis report. For details, see “Displaying data areas” on page 149.
- Information about the effect of the Details option setting on the CICS trace in the real-time or batch reanalysis report has been provided. For details, see “Detail” on page 465.
- Information about setting SLIP traps for certain Fault Analyzer error message has been provided in “Setting Fault Analyzer SLIP traps” on page 229.
- The following message has been added:
  - IDI0182IFor details, see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PI35460 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-06: APAR PI28247 Edition (November 2014)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Permitted the use of &SYSUID. and &TSOPFX. substitution symbols in compiler listing and side file data set names. For details, see “Data set name substitution symbols” on page 460.
- Added information about disabling Fault Analyzer using the IFAPRDxx parmlib member when IBM Problem Determination Modernization Solution Pack for z/OS (5655-DMS) has been installed. For details, see “Turning off Fault Analyzer using the IFAPRDxx parmlib member” on page 355.
- The CICS transaction abend analysis report “CICS levels, Commareas and Channels” display has been changed. For details, see “CICS Levels, Commareas, and Channels” on page 126.
- Added suboption to the RETAINCICSDUMP(AUTO) option to control the suppression of transaction dumps for EXEC CICS DUMP calls. For details, see “RetainCICSDump” on page 497.

This edition of the book is provided in PDF format with the PTF for APAR PI28247 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-05: APAR PI19883 Edition (August 2014)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Changes have been made to the IDIXFXIT user exit parameter list. For details, see “Migrating from V12.1 to V13.1” on page 211. Note that these changes potentially affects not only users migrating from an earlier version of Fault Analyzer, but any users of the IDIXFXIT user exit.
- Information about changing ISPF screen settings have been provided in “Increasing the display area for Fault Analyzer reports” on page 251.
- A new load module type Formatting user exit has been provided. For details, see “The IDIXUFMT load module Formatting user exit” on page 435.
- The ability to toggle hex-dumped storage character translation between EBCDIC and ASCII in the interactive reanalysis report has been provided. For details, see “Displaying storage locations” on page 147.

## **SC19-4116-05: APAR PI19883 Edition (July 2014)**

- Information about a Java API, which can be used to download a Fault Analyzer report, has been provided in Appendix C, “Java API to download Fault Analyzer report,” on page 573.
- The following message has been added:
  - IDI0181WFor details see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PI19883 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-04: APAR PI17012 Edition (June 2014)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- COBOL Explorer, which is an interactive post-mortem debugger for COBOL programs, is provided. For details, see “COBOL Explorer” on page 172
- The CICSDumpTableExclude option is enhanced. For details, see “CICSDumpTableExclude” on page 456
- The CFA command that was issued from the MVS console is enhanced. For details, see “Using CFA from an MVS console” on page 308
- The following messages have been added:
  - IDI0179W
  - IDI0180I

For details, see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PI17012 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-03: APAR PI11055 Edition (April 2014)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Included information about the IPV.SIPVLPA1 library in the Fault Analyzer installation section. For details, see “Making Fault Analyzer modules available” on page 231.
- Information about the need to also use the IBM Problem Determination Tools for z/OS Common Component IPVOPTLM configuration-options module for access to the IPVCNF00 parmlib configuration member, if the Fault Analyzer IDOPTLM configuration-options module is needed for access to the IDICNFxx parmlib configuration member. For details, see “Migrating from V12.1 to V13.1” on page 211 and “Specifying an alternative parmlib data set for IDICNFxx” on page 258.
- A diagram showing Fault Analyzer non-CICS (batch) invocation exit usage, which replaces the previous table-based information, is in Figure 128 on page 226.
- The following messages have been added:
  - IDI0175I
  - IDI0177E
  - IDI0178E

For details, see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PI11055 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-02: APAR PI08456 Edition (January 2014)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Added a statement regarding the use of PDSE version 2 data sets with member generation support. For details, see “Allocating a PDS or PDSE for a history file” on page 261.
- Updated the sample Fault Analyzer configuration file for the IBM Problem Determination Tools for z/OS Common Component Common Server. For details, see “Customize the IBM Problem Determination Tools for z/OS Common Component Common Server” on page 443.
- Added DataSets option data set name symbol substring support. For details, see “Symbol substring specification” on page 462.
- Added information about DB2 V11 support. For details, see “Supported application environments” on page 9.
- The following messages have been added:
  - IDI0173I
  - IDI0174I

For details, see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PI08456 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-01: APAR PM99413 Edition (November 2013)**

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

- Dynamic allocation of the IDIREPRT DDname, using the SYSUDUMP DDname SYSOUT specification, now includes both the SYSOUT class and the form name. For details, see “The real-time analysis report” on page 18.
- Added information about registration of IBM Problem Determination Solution Pack for z/OS in the IDIPRDxx parmlib member. For details, see “Registering Fault Analyzer in the IFAPRDxx parmlib member” on page 238 and “Turning off Fault Analyzer using the IFAPRDxx parmlib member” on page 355.
- The following messages are changed:
  - IDI0030W

For details, see Chapter 36, “Messages,” on page 535.

This edition of the book is provided in PDF format with the PTF for APAR PM99413 in data set IDI.SIDIDOC1(IDIUGPDF).

---

## **SC19-4116-00: October 2013**

This edition of the book provides information applicable to Fault Analyzer Version 13 Release 1.

Changes in this edition include:

- Provided information about migrating from Fault Analyzer V12.1 to V13.1, in “Migrating from V12.1 to V13.1” on page 211.

- Removed chapter "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products". This information is now in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.
- Removed DB2 version 5 section "DB2 stored procedures".
- The Fault Analyzer web interface was changed to use a WebSphere Application Server Liberty profile. For details, see "Using the Fault Analyzer web interface" on page 206 and "Installing the Fault Analyzer web interface" on page 446.
- Provided information about minimum JVM service levels that are required for Java analysis. For details, see "Supported application environments" on page 9.
- Provided information about an optional IDIJVM DD statement. For details, see "Specifying an IDIJVM DD statement" on page 244.
- Provided information about Fault Analyzer's ability to assist with CICS Storage Accounting Area analysis. For details, see "CICS Storage Accounting Area (SAA) overlay assistance" on page 13.
- Added Chapter 17, "Customize Fault Analyzer by using an IDIOPTLM configuration-options module," on page 257.
- The following messages have been changed:
  - IDI0154W

For details, see Chapter 36, "Messages," on page 535.

Following the product install, this version of the book is available in PDF format in the data set IDI.SIDIDOC1(IDIUGPDF).

---

## **Part 1. Using Fault Analyzer**



---

# Chapter 1. Introduction

This introduction outlines the analysis process, and describes other features of IBM Fault Analyzer for z/OS (Fault Analyzer).

The purpose of Fault Analyzer is to determine why an application abends. After analyzing information about the application and its environment, Fault Analyzer generates an analysis report. The report describes the problem in terms of application code, which means that application developers and maintainers are not forced to interpret a low level system dump or system-level error messages. As a result, the reason for the abend is made available sooner and with less effort.

---

## The analysis engine

The core of Fault Analyzer is its purpose-built analysis engine. This engine is brought into play whenever analysis is required. Analysis is automatic after an abend, application-initiated for the program SNAP interface, or user-initiated for fault reanalysis.

The analysis engine is an expert system that encapsulates the collective debugging experience of leading IBM software architects, developers, and testers.

---

## The analysis process

The Fault Analyzer analysis process begins with a real-time analysis, caused by either an abend or the explicit call to the SNAP interface, followed by a reanalysis if necessary.

The processes involved in the different types of analysis functions capable of being performed by Fault Analyzer are discussed in the following.

### Real-time abend analysis

When a program abends, the abend processing (MVS or subsystem) is intercepted and Fault Analyzer is automatically invoked via an appropriate exit for the processing environment. See “Exits for invoking Fault Analyzer” on page 223 for detailed information about the types of exits available.

Fault Analyzer performs fault analysis processing, and then records details about the abend in a history file. Fault Analyzer writes the fault analysis report to the job, and a summary to the SYSLOG. It also saves the analysis report in the history file along with a minidump consisting of a copy of all virtual storage pages that were referenced during the analysis process. This mode of operation is known as “real-time analysis”. Figure 2 on page 4 illustrates this process.

## The analysis process

---

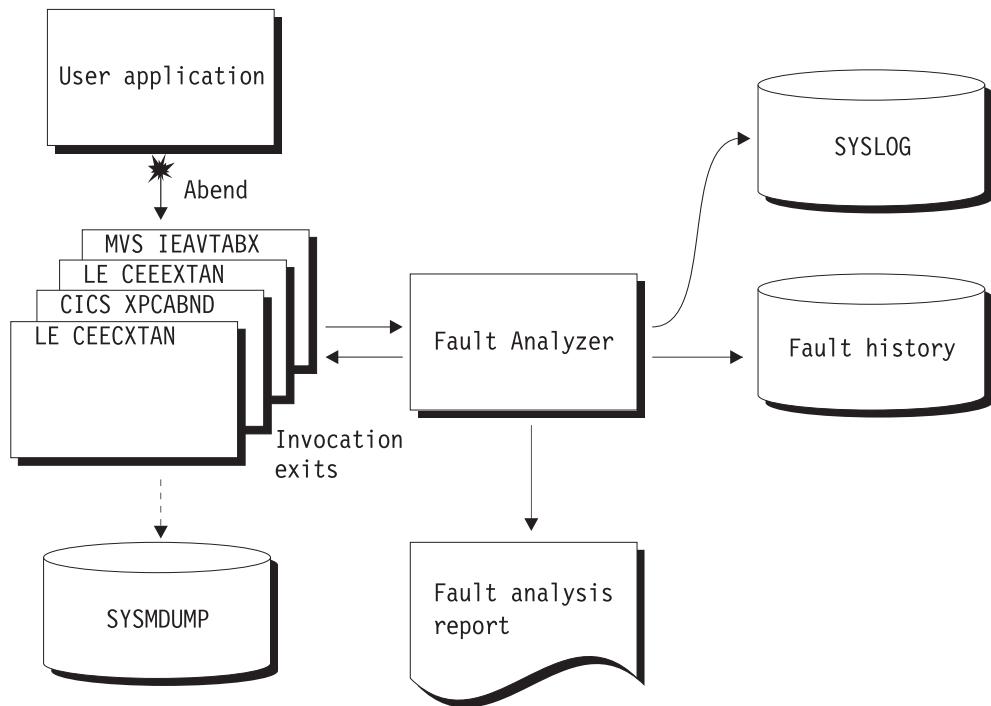


Figure 2. Real-time abend analysis

For an example of a real-time fault analysis report, see “Sample reports” on page 204.

Fault Analyzer is designed to only be invoked for the first abend under a given TCB.

If Fault Analyzer deems the analysis to be successful, and either a SYSMDUMP, SYSABEND, or SYSUDUMP was specified for the abending job step, then Fault Analyzer tells MVS to suppress the dump.

The minidump is written to the history file for any real-time invocation of Fault Analyzer, without the need to specify any options or make changes to JCL. There are three exceptions when the minidump is suppressed:

- The number of pages exceed an installation-defined limit (see “MaxMinidumpPages” on page 482)
- The fault is a duplicate of another fault (see “NoDup” on page 482)
- An End Processing user exit requested the minidump to be suppressed (see “End Processing user exit” on page 393).

The minidump permits reanalysis of faults that occurred in any processing environment, regardless of whether a SYSMDUMP was also written.

If the application does not abend, then Fault Analyzer consumes no processing resource. This parsimony makes Fault Analyzer equally suited for application development, testing, or production environments.

Instead of forcing application developers and maintainers to interpret a low-level system dump or system-level error messages, the fault analysis report describes the

fault in terms of the application code. Where possible, the report quotes the source statement where the abend occurred and, for COBOL and PL/I, the names and values of data items used in the statement.

Fault Analyzer is supplied with softcopy versions of many manuals from the OS/390 On-line Library. Fault Analyzer extracts message and abend code descriptions, as well as other relevant information, from these manuals, and inserts them into the analysis report where applicable. In some cases, Fault Analyzer supplies its own message and abend code descriptions. These provide more specific information than the descriptions found in the manuals.

Generally, when the associated compiler listing (or side file) of the abending program is available on-line, then application abends are isolated down to the program source statement involved in the abend (see "Compiler listing or side file selection criteria" on page 12 for information about the criteria used when selecting compiler listings or side files). When the listing is not available, the problem is diagnosed down to program name and offset, with disassembly of the machine instructions.

For much of the time, the analysis report provided by Fault Analyzer and written to the job output is adequate, and you do not need any further fault information for successful problem determination. However, if you do want to extract more information about the abend, you can ask for a reanalysis of the fault, using the ISPF interface to initiate batch or interactive reanalysis.

### Real-time SNAP analysis

There is basically no difference between the Fault Analyzer real-time abend analysis process and the real-time SNAP analysis process, except for the way in which Fault Analyzer is invoked.

The program SNAP interface permits an application program to invoke Fault Analyzer by including the appropriate calls where desired. This way, the application programmer can obtain an analysis of the current environment in situations where the application might not abend. The call to Fault Analyzer is non-disruptive to the application program, which is able to continue execution following the analysis.

An illustration of the real-time SNAP analysis process is provided in Figure 3 on page 6.

## The analysis process

---

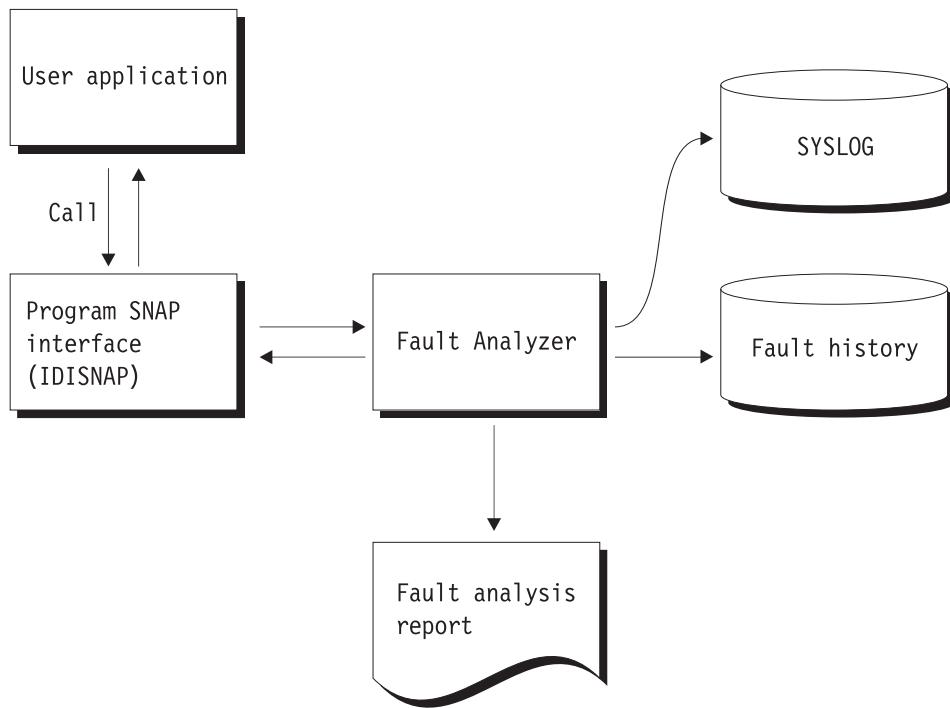


Figure 3. Real-time SNAP analysis

---

## Fault reanalysis

The reanalysis process is essentially identical to the real-time analysis process, except for the following:

- Fault Analyzer obtains the required information from the saved minidump (and/or SYSMDUMP) instead of the abending program's virtual storage.
- The history file is not updated.
- No summary is written to the SYSLOG.

Figure 4 on page 7 illustrates the reanalysis process.

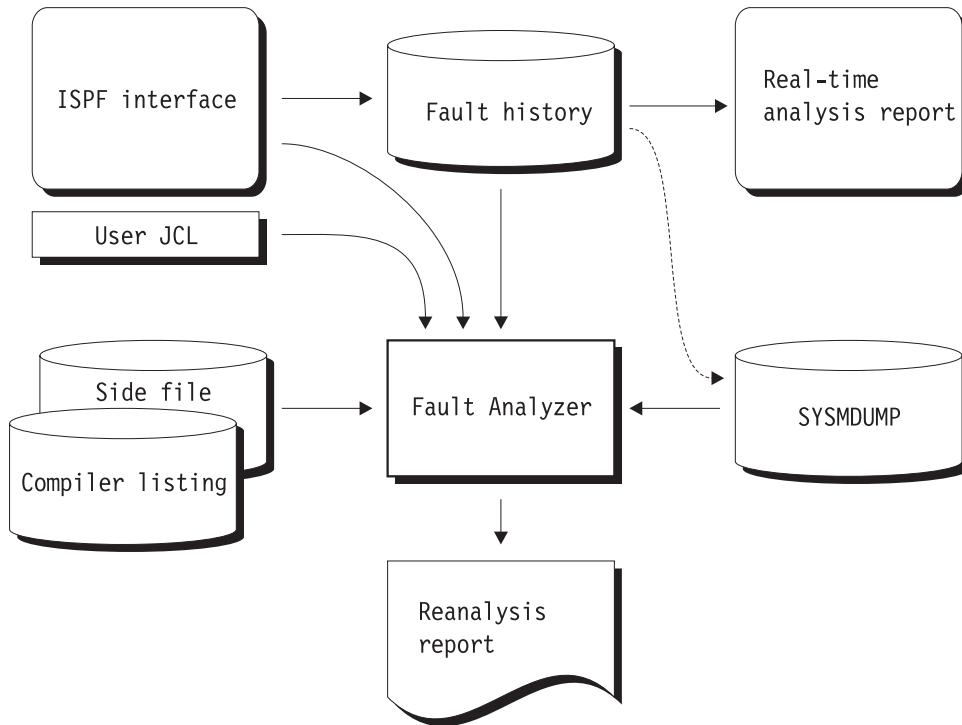


Figure 4. Reanalysis

If a SYSMDUMP data set is available, then the fault history entry holds the dump data set name in case it is required during reanalysis or user-selected display of storage locations that are not included in the minidump. If the dump data set is not available, then you can still perform reanalysis using the minidump which is included in the history file entry. Only if neither a minidump, nor a SYSMDUMP data set exists for a given fault, then you cannot initiate reanalysis, but you can still look at the real-time analysis report from the ISPF interface.

For the reanalysis, you can make a listing (or a side file) available, if one was not available when the real-time analysis was performed. Fault Analyzer is now able to provide the source statement information relevant to the abend.

Fault reanalysis can be performed in two different ways: batch or interactive. Both methods can be initiated using the Fault Analyzer ISPF interface.

### Batch reanalysis

The format of the batch reanalysis report is the same as the real-time analysis report. The batch reanalysis report is written as a sequential file to a DD statement in the reanalysis job but it is not saved in the fault history file entry.

As well as using the ISPF interface to initiate batch reanalysis, you can also submit a batch reanalysis job using your own JCL. See Chapter 3, “The Fault Analyzer ISPF interface,” on page 33 for more information.

### Interactive reanalysis

Unlike the real-time analysis report and the batch reanalysis report, both of which are written as sequential files, the interactive analysis report is presented as a series of panels that lets you choose the sections of interest.

## The analysis process

Further, it lets you view the contents of storage areas included in the minidump and/or SYSMDUMP data set, but not necessarily formatted out in the report. It is also the only way to perform CICS system abend analysis.

Interactive reanalysis can only be initiated using the Fault Analyzer ISPF interface.

---

## Fault history files

Fault history files are PDS(E) data sets that contain information about faults that have been analyzed by Fault Analyzer. Fault entries, which are stored as separate members in the history file, contain the following type of information:

- Real-time analysis key information, such as abend code and failing program name
- Execution environment details, such as job name, system ID, and date and time when the fault occurred
- Associated real-time analysis report (if applicable)
- Saved minidump (if applicable)
- Name of associated SYSMDUMP or SVC dump data set (if applicable)

Each new fault entry in a history file is given an identifier, which is unique to that history file. The ID consists of a 1-3 character prefix, followed by a 5-digit sequence number. The default prefix is "F", but this default can be changed using the IDIUTIL batch utility (see Chapter 27, "Managing history files (IDIUTIL utility)," on page 337 for details of this utility).

The sequence numbering starts at 1 and is incremented by 1 for each new fault entry created. When the sequence number reaches 99999, it wraps back to 1. If a given fault ID already exists, then the next available fault ID is assigned.

The maximum number of fault entries that can be contained in a history file can be set using the IDIUTIL batch utility.<sup>1</sup> Setting this limit has no impact on the assignment of sequence numbers to new fault entries created. The default is not to limit the number of fault entries.

Although a history file can be either a PDS or a PDSE data set, PDSE data sets are recommended because they provide concurrent member write capability which is not possible with PDS data sets. Fault Analyzer provides better sharing and performance when using PDSE history files.

You can view fault entry details, browse the real-time analysis report, start batch and interactive fault reanalysis, or delete fault entries from an interactive display of a history file. These actions are described in Chapter 3, "The Fault Analyzer ISPF interface," on page 33.

A view member can be created in a data set identified by the IDIVIEWS DDname that contain the names of any number of history files that you want to display simultaneously using the Fault Analyzer ISPF interface. For details, see "Using views" on page 37.

---

1. This value can be exceeded if the maximum fault limit has been reached, and all fault entries have been "locked". For details about fault entry locking, see "Viewing fault entry information" on page 82.

## Special members in the history file data set

If listing the members of a history file data set with ISPF or similar, you might notice members whose names start with "##". These are not fault entries, but instead internal control members used by Fault Analyzer:

### ##INDEX

This member contains an index of all fault entries in the history file and is used for quick access to basic information for each fault. It is also the sole repository for all duplicate information against any fault in the history file.

If the ##INDEX member is missing for any reason, then it is rebuilt the next time the history file is updated. This situation might happen, for example, when real-time analysis causes a new fault entry to be created, or when user-information for an existing fault entry is updated.

**Note:** The rebuilt ##INDEX member does not contain any information about duplicate fault occurrences.

A sample assembler program, which can be used as the basis for user-specific reporting or statistical analysis of a history file ##INDEX member, is available as member IDIS\$NDX in data set IDI.SIDISAM1.

### ##BACKUP

This member contains a copy of the history file specific settings that have been set in the ##INDEX member by using the IDIUTIL batch utility SetFaultPrefix, SetMaxFaultEntries or SetMinFaultEntries control statements (for details, see Chapter 27, “Managing history files (IDIUTIL utility),” on page 337), or by using the Fault Analyzer ISPF interface (for details, see “Change fault history file settings” on page 57). If, for any reason, the ##INDEX member information is lost, then these settings are recovered from the ##BACKUP member.

---

## Supported application environments

Fault Analyzer supports applications running under z/OS® and OS/390® in the following applications environments:

- COBOL (see Table 1 on page 10 for details)
- PL/I
- Assembler
- C/C++
- Language Environment
- UNIX System Services
- CICS® (see Table 1 on page 10 for details)
- IMS™ (see Table 1 on page 10 for details)
- DB2® (see Table 1 on page 10 for details)
- MQSeries® (see “MQSeries support” on page 11 for details)
- Java™ (see Table 1 on page 10 for details)

In the z/OS environment, Fault Analyzer executes in 31-bit addressing mode and performs analysis on 24-bit or 31-bit addressing mode applications. Multithread, DLL, and XPLink applications are supported.

| Fault Analyzer performs only limited analysis of 64-bit addressing mode applications.

| The 64-bit support includes, for some events:

- | • Recognition of 64-bit addressing mode (AMODE 64)

## Supported application environments

- 64-bit machine instruction operand addresses
- 64-bit general purpose registers

The 64-bit support does not include:

- Associated storage areas above the bar, beyond 4K around general purpose registers when using the Detail(Long) option.

C++ support is basic and does not provide any class information.

Only execution in home-space mode is supported.

Real-time analysis is limited to tasks running in TCB protect key 8 for non-CICS and key 8 or 9 for CICS. Any other protect key results in message IDI0123S being issued. To still use Fault Analyzer to analyze abends in such tasks, a SLIP trap can be set on the IDI0123S message to capture an SVC dump which can then be analyzed via the ISPF interface action-bar "File" pull-down menu option 5.

*Table 1. Application environments supported by Fault Analyzer*

		Fault Analyzer		
		V13.1	V12.1	V11.1
CICS/TS	V4R1 and earlier	GA	GA	GA
	V4R2	GA	GA	UK68814
	V5R1 ( <b>Note 1</b> )	GA	UK83720	UK83847
	V5R2 ( <b>Note 1</b> )	UI18641	UI18643	UI18644
Enterprise COBOL	V4R1 and earlier	GA	GA	GA
	V4R2	GA	GA	GA
	V5R1	GA	UK96928	-
Enterprise PL/I	V3R8 and earlier ( <b>Note 2</b> )	GA	GA	GA
	V3R9	GA	GA	GA
	V4R1	GA	GA	UK65276
DB2	V9 and earlier	GA	GA	GA
	V10	GA	GA	GA
	V11	UI13465	UI13466	UI13511
IMS	V11 and earlier	GA	GA	GA
	V12	GA	GA	GA
	V13	GA	UK93704	UK93741
IBM® Java	Version 6.0.1 SR6 ( <b>Note 3</b> )	GA	UK98036	UK98099
	Version 7 SR5 ( <b>Note 3</b> )	GA	UK98036	UK98099
z/OS	V1R10 and earlier	GA	GA	GA
	V1R11	GA	GA	GA
	V1R12	GA	GA	GA
	V1R13	GA	GA	UK70024
	V2R1	GA	UK95217	UK95249
	V2R2	UI31227	UI31231	UI31220

Table 1 on page 10 notes:

- 1 Excludes 64-bit application support.
- 2 The minimum level of Enterprise PL/I required for complete Fault Analyzer source level support is version 3 release 2 with PTFs UQ71704 and UQ71690 installed.
- 3 Java support:
  - Earlier versions or levels of Java might not be supported by Fault Analyzer.
  - Java support under CICS is limited to CICS V4R1 or earlier.

Table 1 on page 10 legend:

Symbol	Meaning
<b>GA</b>	General availability. This version is the base version of Fault Analyzer without any maintenance installed.
<b>UInnnnn/UKnnnnn</b>	Earliest Fault Analyzer PTF level providing support.
<b>PInnnnn</b>	Earliest Fault Analyzer APAR level providing support. - No support is available on this version of Fault Analyzer at any maintenance level.

## MQSeries support

MQSeries support consists of one of the following:

### Abends occurring in a MQSeries call

In addition to normal information about the abend itself, the MQSeries API description is provided.

### Information about prior calls to MQSeries

The information consists of identification of the last MQSeries call (in ascending source line order) that resulted in a non-zero reason code, given the current content of the reason code data field used in the call.

The reason code is provided, along with its explanation.

To facilitate this information, the following conditions must be met:

- MQSeries static linkage is used.
- The application issuing the MQSeries call is written in COBOL.
- Compiler listing or side file is provided.

---

## Binder-related dependencies

In order to map CSECTs in load modules, Fault Analyzer calls the IBM Binder program by means of its Application Programming Interface. This call is generally done during real-time analysis, but might also be done during interactive analysis of CICS system dumps. Since the Binder supports load modules residing in PDS(E) data sets only, then Fault Analyzer is not able to identify CSECTs in load modules that have been loaded from any other type of storage.

---

## **Setting up existing programs for fault analysis**

You do not need to make any changes at all to existing programs to allow Fault Analyzer to produce an analysis of any fault (provided that you install the USERMOD described in “Eliminating the need for a dump DD statement (++IDITABD)” on page 255 for batch jobs). Nor do you need to recompile programs. However, if you store compiler listings or side files in the appropriate repository, then Fault Analyzer is able to identify the source statement of the abending program. (If you choose to not store listings or side files, you can still provide one after an abend has occurred. This approach makes it possible for Fault Analyzer to extract more information when you perform reanalysis.)

To provide a side file, you might need to recompile your programs, since appropriate side files are only produced when certain compiler options are requested. If you already have compiler listings that were produced with the correct compiler options, you can create side files without needing to compile again. The advantage of the side file is that it is more compact than a listing. For more information, see Chapter 20, “Providing compiler listings or Fault Analyzer side files,” on page 285.

### **Additional region size required**

Fault Analyzer runs in the same region as your abending program at the time of the abend. Therefore, there must be spare GETMAIN storage that is not used by the application in order for Fault Analyzer to run and analyze the program storage in its abend state. Initially, up to 16 megabytes of storage might be required, depending on the execution environment. This extra region size increases as the size and complexity of the abending program increases.

For detailed information about storage requirements, see “Storage recommendations” on page 222.

In situations where Fault Analyzer is unable to obtain sufficient storage for the real-time analysis of a fault, a SYSMDUMP can be taken for subsequent batch or interactive reanalysis.

---

## **Compiler listing or side file selection criteria**

Fault Analyzer basically performs two types of check when selecting a compiler listing or side file to be used for source-level analysis:

1. A size check is performed, which varies from language to language, where an attempt is made to match the size and contents of the load module with the compiler listing or side file. For example, when the COBOL compiler LIST option is used, the size checks include matching the offset and contents of the last 12 assembler instructions in the CSECT. Also for the current COBOL compilers, the working storage size and TGT size are also checked.
2. A date and time check is performed between the load module and the compiler listing or side file. Provision is made for compiler listings being created after the date and time that is associated with the load module.

To obtain detailed information about why a particular compiler listing or side file was selected or rejected, the IDITRACE facility can be used. See “IDITRACE information” on page 292 for details on how to use this facility.

---

## **Special processing of Language Environment CEEWUCHA abends**

Language Environment provides a sample user condition handler, CEEWUCHA, that you can use to alter the default behavior of LE to get behavior that is similar to VS COBOL II. This condition handler is specified by using the LE USRHDLR(CEEWUCHA) option.

Fault Analyzer suppresses an initial U4038 LE abend to provide an Event Summary with the CEEWUCHA abend code as the important abend code when these two conditions are true:

- The LE USRHDLR(CEEWUCHA) option is in effect for an application abend that is analyzed by Fault Analyzer.
- LE passed control to this condition handler.

If a condition handler with any name other than CEEWUCHA is specified in the LE USRHDLR option, the initial LE user abend is not suppressed.

---

## **WTO routing and descriptor codes used by Fault Analyzer**

All write-to-operator messages (WTOSs) issued by Fault Analyzer specify routing code 11 (Programmer Information) and descriptor code 7 (Task-Related).

---

## **CICS Storage Accounting Area (SAA) overlay assistance**

Fault Analyzer provides two features designed to assist with CICS SAA (Storage Accounting Area) overlays:

- Concurrent CICS task storage information

A display is available from the interactive reanalysis report, which cannot only be used to identify overlays in the current transaction's storage, but also includes relevant storage areas owned by other transactions which might have been involved in the overlay.

For details, see "CICS Transaction Storage Summary" on page 120.

- Early detection of SAA overlay by implementing an XEIN global user exit

The XEIN exit is used to ensure that if a storage overlay is detected by CICS, that either Fault Analyzer is invoked to perform normal real-time analysis, or an SVC dump is taken which can later be reanalyzed.

For details, see "Implementing an XEIN global user exit" on page 314.

## **WTO routing and descriptor codes used by Fault Analyzer**

---

## Chapter 2. Real-time analysis

Real-time analysis occurs when an application abends and Fault Analyzer is invoked through one of the supplied invocation exits (see “Exits for invoking Fault Analyzer” on page 223), or a call to the program SNAP interface is made, and the job has not been excluded from analysis.

Generally, real-time analysis produces two things:

- A report, which is written to JES by default (see “The real-time analysis report” on page 18).
- A fault entry in a history file, which provides the ability to perform reanalysis of the fault.

A copy of the report that is written to JES is also included in the fault entry, and can be viewed from the ISPF interface. You cannot change the report by setting options to different values at the time you view it. If you want to look at more (or less) detail, you must reanalyze the fault with adjusted options or a supplied listing or side file.

This step is the first step in the fault analysis process. In most cases, the analysis is deemed satisfactory, and you do not need to reanalyze the fault.

For a particular job you can adjust some options before you run the job.

All virtual storage pages that were referenced during the analysis in the abending task's address space are written to the history file as a minidump, unless the MaxMinidumpPages option in effect specifies a lower limit.

**LOADER restriction:** Fault Analyzer does not work correctly if using the LOADER (IEWBLDGO) since the load-and-go technique of link-editing modules does not write them to a data set. The data set copy of the load module is needed in order to determine CSECT names, lengths, and starting offsets.

---

## Dump suppression

The types of dumps that can be suppressed are:

- SYSABEND, SYSUDUMP, or SYSDUMP (if Fault Analyzer was invoked using the IEAVTABX MVS change options/suppress dump exit, IDIXDCAP).
- CICS transaction dumps.

**Note:** The suppression of CICS transaction dumps requires Fault Analyzer to be installed in either the XPCABND or the XDUREQ CICS Global User Exit (see Chapter 21, “Customizing the CICS environment,” on page 301).

Suppression of dumps upon *successful completion*<sup>2</sup> of analysis is the default.

However, when using

```
EXEC CICS DUMP TRANSACTION DUMPCODE(xxxx)
```

---

2. Successful completion of analysis is defined as Fault Analyzer having reached the normal end of analysis, without having abended or issued any S-level messages.

## Dump suppression

under CICS, the transaction dump is only suppressed if this was requested prior to Fault Analyzer being invoked, for example by an earlier exit. This is due to the SUPPRESSED response being passed back to the issuing application program, which if not handled correctly can lead to AEXW abends. See “Fault Analyzer and CICS global user exits” for more information on transaction dump suppression and CICS global user exits.

To override the default dump suppression, use either:

- The RetainCICSDump(ALL) option for CICS transaction faults (see “RetainCICSDump” on page 497).
- The RetainDump(ALL) option for non-CICS transaction faults (see “RetainDump” on page 498).
- An End Processing user exit (see “End Processing user exit” on page 393).

Dump suppression should not be confused with suppression of analysis or suppression of fault entry creation (see “Real-time exclusion processing” on page 27).

## Fault Analyzer and CICS global user exits

If Fault Analyzer is installed in the CICS XPCABND global user exit (GLUE), and transaction abend analysis completes successfully (either an LE or non-LE application), then Fault Analyzer passes the appropriate return code to CICS to suppress the transaction dump. The implication of this passing is that any subsequent dump-related CICS GLUE, for example XDUREQ, is then not invoked by CICS.

If your environment is such that you require subsequent CICS dump GLUEs to always be called, or if you require CICS transaction dumps to always be written, you should do one of the following:

- Code a Fault Analyzer End Processing user exit and set the ENV.SUPPRESS\_DUMP flag to 'N'.
- Set the RetainCICSDump option to ALL—see “RetainCICSDump” on page 497 for more details.

---

## Fault history file selection

When a fault is being analyzed in real time by Fault Analyzer, a history file must be available in which details of the analysis can be recorded.

There are a number of ways in which the name of the history file can be provided to Fault Analyzer. The following is a list of these in the order of their override significance (each entry in the list overrides all previous entries):

1. The product default name, IDI.HIST.
2. The IDIHIST suboption of a DataSets option that is specified in the parmlib config member, IDICNF00. This information includes either the logical parmlib concatenation or the installation-wide alternate parmlib data set name provided via the IDISCNF USERMOD supplied with Fault Analyzer.
3. The IDIHIST suboption of a DataSets option that is specified in a config member identified via the IDICNFUM user options module.

**Note:** If a user options module is used, it replaces the default IDICNF00 parmlib config member. Thus, even if the user options module designated

config member did not include an IDIHIST suboption of a DataSets option, any specification of IDIHIST in the default IDICNF00 parmlib config member would not be recognized.

4. The IDIHIST suboption of a DataSets option provided via the IDIOPTS DDname in the abending job step.
5. An explicitly coded IDIHIST DD statement in the abending job step.
6. The data set name provided by an Analysis Control or End Processing user exit in the ENV data area IDIHIST field.

---

## Controlling the real-time analysis with options

Set options globally, so they control the output for all jobs. However, you can also set an option just for one job. In this case, you should set the option in the user options file via the IDIOPTS DDname.

See Chapter 33, “Options,” on page 451 for information about all available options and the different ways in which they can be specified.

Options that you are more likely to use for real-time analysis are:

**RetainDump(ALL)**

Specify this option if you want to retain the SYSABEND, SYSUDUMP, or SYSMDUMP unconditionally. Fault Analyzer permits the writing of the dump, and records the name of the dump data set in the history file if a SYSMDUMP DD statement was specified. Without this option, many dumps are suppressed when Fault Analyzer deems that the analysis it has performed is adequate. This option does not affect the writing of the minidump to the history file. See “RetainDump” on page 498 for more details.

The dump disposition part of this option is applicable to the use of the MVS IEAVTABX change options/suppress dump exit, IDIXDCAP, only.

**Detail** Specify this option if you want to adjust the level of detail given in the real-time analysis report. (If a dump is produced, you can change this option when you perform a reanalysis.) See “Detail” on page 465 for more detail.

**Exclude**

Specify this option if you want to exclude this job from analysis. See “Exclude/Include” on page 469 for more detail.

**NoDup**

Specify this option if you want to change the way that duplicate faults are handled by default. See “NoDup” on page 482 for more detail.

**CICSDumpTableExclude**

Specify this option if you want to exclude CICS transaction fault analysis using the CICS transaction dump code table. See “CICSDumpTableExclude” on page 456 for more detail.

You can also use the DataSets option to point to listings and side files. See “DataSets” on page 457 for more detail.

### Pointing to listings with JCL DD statements

No DD statements are required to run Fault Analyzer in either batch or real time. However, for versions of z/OS prior to z/OS 2.2, a SYSMDUMP, SYSUDUMP or SYSABEND DD statement is needed for normal MVS dump processing in real time when using the MVS IEAVTABX change options/suppress dump exit, IDIXDCAP, unless the IDITABD USERMOD is applied (see “Eliminating the need for a dump DD statement (++IDITABD)” on page 255 for details about the effect of this USERMOD).

You can specify the following DD statements in the JCL if appropriate. If they are not specified, the definitions from the PARMLIB configuration member IDICNF00, the IDIOPTS user options file, or an Analysis Control user exit are used to identify these data sets:

**IDILC** PDS(E) data set containing C compiler listings

**IDILCOB**

PDS(E) data set containing COBOL compiler listings (other than OS/VS COBOL)

**IDILCOBO**

PDS(E) data set containing OS/VS COBOL compiler listings

**IDISYSDB**

PDS(E) data set containing COBOL or Enterprise PL/I SYSDEBUG, or XL C/C++ MDBG side files.

**IDILPLI**

PDS(E) data set containing PL/I compiler listings (other than Enterprise PL/I)

**IDILPLIE**

PDS(E) data set containing Enterprise PL/I compiler listings

**IDIADATA**

PDS(E) data set containing SYSADATA from Assembler compilations

**IDILANGX**

PDS(E) data set containing LANGX side files for all languages

Do not specify a member name on any of the above DD statements. See Chapter 20, “Providing compiler listings or Fault Analyzer side files,” on page 285 for further details on the use of these files.

### The real-time analysis report

The real-time analysis report is produced whenever Fault Analyzer analyzes an abend, or is invoked by IDISNAP or the equivalent com.ibm.fa.snap.Dump Java class, unless the DeferredReport option is used (see “DeferredReport” on page 463) or the report is suppressed (see “Suppressing real-time reports” on page 19). The report is written to the IDIREPRT DDname, which is dynamically allocated to SYSOUT=*class*, if no prior allocation exists, and thus is included as part of the normal job output on the JES spool.

The SYSOUT class used (*class*) is the default job output class (SYSOUT=\*), or if a SYSUDUMP DD statement in the abending job step specifies a JES SYSOUT class, then the same output class and form name are used for the Fault Analyzer real-time report.

If you want to divert the real-time analysis report to another file, then adjust the DD card as required. For example:

```
//IDIREPRT DD DISP=(,CATLG),DSN=MY.REPORT.DS,  
//           DCB=(RECFM=VB,LRECL=137),SPACE=(CYL,(1,1))
```

Alternatively, a user exit can be used to allocate IDIREPRT to a different output class—for details, see “Controlling the SYSOUT class of real-time reports.”

The IDIREPRT DDname is opened with LRECL=137. Any existing data set attributes must be compatible with this logical record length.

The IDIREPRT allocation for CICS transaction abends is the same as for any other type of abend.

See Chapter 9, “The Fault Analyzer report,” on page 197 for general information about the contents of the Fault Analyzer report, and for report examples.

## Combining Fault Analyzer real-time reports

By default, all real-time reports are written to separate JES spool files. This process is generally considered advantageous for subsystems, such as CICS, IMS message-processing regions, or WLM-managed DB2, where multiple reports can be expected written before the subsystem is restarted.

If, for any reason, you prefer writing the reports to a single spool file, then add an IDIREPRT DD statement to the job or startup procedure, for example:

```
//IDIREPRT DD SYSOUT=*
```

## Controlling the SYSOUT class of real-time reports

If no IDIREPRT allocation already exists, then Fault Analyzer dynamically allocates IDIREPRT to SYSOUT=\*, or to the same SYSOUT class as the SYSUDUMP DDname. Change to a different SYSOUT class by adding a DD statement to the job or startup procedure, for example:

```
//IDIREPRT DD SYSOUT=sysout-class
```

Alternatively, an Analysis Control user exit can be used to allocate IDIREPRT to a required class, as shown in the following REXX example:

```
/* REXX */  
/*****************************  
/* Sample Fault Analyzer Analysis Control user */  
/* exit to allocate IDIREPRT to SYSOUT class F. */  
/*****************************  
"IDIALLOC DD(IDIREPRT) SYSOUT(F)"  
exit 0
```

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))  
Exits(CONTROL(REXX(ABC)))
```

## Suppressing real-time reports

To suppress the writing of any Fault Analyzer reports to the JES spool, you can add the following DD statement to the job or startup procedure:

```
//IDIREPRT DD DUMMY
```

## The real-time analysis report

Alternatively, an Analysis Control user exit can be used to allocate IDIREPRT to DUMMY, as shown in the following REXX example:

```
/* REXX */
/*****************************************/
/* Sample Fault Analyzer Analysis Control user */
/* exit to suppress the analysis report. */
/*****************************************/
"IDIALLOC DD(IDIREPRT) DUMMY"
exit 0
```

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(CONTROL(REXX(ABC)))
```

Note that the real-time report is able to be written to the history file, regardless of the above suppression of the JES spool report, and can be viewed from there using the Fault Analyzer ISPF interface.

In a CICS environment, it is preferable to use the DeferredReport option instead, which is also the default. For details, see “DeferredReport” on page 463.

---

## The SYSLOG summary

During real-time analysis, a message is written to the operator console providing a one-line summary of the fault reason.

Following is an example of such a message:

```
IDI0002I There was an unsuccessful REWRITE of file MYFILE01 (file status 44)
in program COBFERRD at line # 21
```

If the Quiet option is in effect, then this message might not be written to the SYSLOG.

---

## Using the program SNAP interface (IDISNAP)

A program SNAP interface is provided to assist users in debugging problems with applications that do not abend, or that for any other reason cannot be analyzed by Fault Analyzer using one of the normal abend invocation exits described in “Exits for invoking Fault Analyzer” on page 223. This interface permits a call to Fault Analyzer from anywhere within an application program to request an analysis of the current environment. The program SNAP interface module name is IDISNAP.

An example of where a call to IDISNAP might be used is in a DB2 application after execution of an SQL statement that results in a negative SQLCODE.

Apart from the way in which Fault Analyzer is invoked, there is no difference between this type of analysis and any other real-time analysis caused by an abend.

It is recommended that you invoke IDISNAP dynamically to ensure that you are always using the most current version:

- If the invoking program is compiled with the DLL option, then invocations of IDISNAP need to be STATIC rather than DYNAMIC.
- For programs written in C, IDISNAP can only be invoked dynamically.

### IDISNAP invocation

The following describes the entry and return specifications for IDISNAP.

#### Entry specifications

On entry to IDISNAP, the contents of registers must be:

##### Register

##### Contents

- 1** Zero, or address of input parameter list (see below).
- 13** Address of 72-byte register save area.
- 14** Return address.
- 15** Entry point address of IDISNAP.

**Input parameter list:** Unless R1 is zero, then register 1 must contain the address of a parameter list, consisting of a list of addresses (OS linkage). Each address in the parameter list points to a parameter.

*Table 2. IDISNAP input parameters*

Parameter	Number of bytes	Description
Parameter 1	4	<p>Specifies the number and type of optional parameters that follow in the format:  <i>t</i>  <i>nnn</i></p> <p>where:</p> <p><i>t</i>      Specifies the format of parameter 3 as either:</p> <ul style="list-style-type: none"> <li><b>0</b>      To indicate that a fixed-length 140 byte parameter area is used.</li> <li><b>N</b>      To indicate that a variable-length null-terminated parameter area is used.</li> </ul> <p><i>nnn</i>      Specifies the number of parameters that follow parameter 1 as either:</p> <ul style="list-style-type: none"> <li><b>000</b>      To indicate that only parameter 1 is specified.</li> <li><b>001</b>      To indicate that parameters 1 and 2 are specified.</li> <li><b>002</b>      To indicate that parameters 1, 2, and 3 are specified.</li> <li><b>00V</b>      To indicate that parameters 1, 2, and 3 are specified, as well as one or more storage range address pairs in subsequent parameters.</li> </ul> <p><b>Note:</b> Specifying 0000 in this parameter is the equivalent to invoking IDISNAP with register 1 set to zero.</p>

## Using the program SNAP interface (IDISNAP)

*Table 2. IDISNAP input parameters (continued)*

Parameter	Number of bytes	Description
Parameter 2	140	<p>First 40 bytes is a user title, the remainder is reserved for use by Fault Analyzer. Any unused portion of the user title should be set to blanks.</p> <p>The specified user title is added to the heading of the real-time report, and can be viewed and updated from the Fault Analyzer ISPF interface Fault Entry List display.</p> <p>Parameter 1 must be 0001 or 0002 in order for this parameter to be processed.</p>
Parameter 3	140 if parameter 1 is 0002, or varying if parameter 1 is N002.	<p>Fault Analyzer options. For example, to prevent the creation of a history file fault entry, specify the DATASETS(IDIHIST(NULLFILE)) option, or to request that specific areas of storage are to be shown in the analysis report, specify the StorageRange option (see "StorageRange" on page 501 for details).</p> <p>Parameter 1 must be either 0002, 000V, N002, or N00V in order for this parameter to be processed:</p> <ul style="list-style-type: none"> <li>If parameter 1 is 0002, then the area pointed to by this parameter must be 140 bytes, with any unused portion set to blanks.</li> <li>If parameter 1 is N002, then the area pointed to by this parameter is a character string of varying length, which must be delimited by a null-character (X'00').</li> </ul>

The following parameters are for pairs of storage range begin and end addresses. The maximum number of address ranges that can be specified is 160.

Using the address range parameters is an alternative to, and overrides, specification of the StorageRange option in parameter 3.

To use these extra parameters, parameter 1 must be either 000V or N00V:

- If parameter 1 is 000V, then the area pointed to by parameter 2 must be 140 bytes, with any unused portion set to blanks.
- If parameter 1 is N00V, then the area pointed to by parameter 2 must be 1 - 1024 bytes, with a null-character (X'00') immediately following the last character in the options string.

Parameter <i>n</i>	4	Storage range begin address.
Parameter <i>n+1</i>	4	Storage range end address.

Options that are specified in parameter 3, or implicitly via the storage range address pairs starting with parameter 4, are passed as PARM field options to the main Fault Analyzer analysis module, IDIDA.

### Return specifications

On return from IDISNAP, the contents of registers are:

Register	Contents
----------	----------

- |      |            |
|------|------------|
| 0-1  | Undefined. |
| 2-14 | Unchanged. |

15      Zero.

### Example 1 (COBOL)

The following is an example of a COBOL program calling IDISNAP five times, showing each of the different invocation styles. As indicated by the DYNAM option in the CBL statement of the COBOL source, IDISNAP is in this example being called dynamically.

```
CBL APOST,NOOPT,DYNAM,XREF,LIST,SSRANGE,RENT,MAP
IDENTIFICATION DIVISION.
PROGRAM-ID. COBMST4X
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
FILE SECTION.

WORKING-STORAGE SECTION.
01 FILLER      PIC X(20) VALUE 'WORKING-STORAGE'.
01 PARM1       PIC X(4) .
01 PARM2.
02 PARM2MSG    PIC X(40) VALUE 'HEADING FOR IDIXSNAP'.
02 PARM2WORK   PIC X(100) .
01 PARM3       PIC X(140) VALUE 'DATASETS(IDIHIST(NULLFILE))'.
01 DATAA       PIC X(200) VALUE 'DATAA'.
01 DATAB       PIC X(200) VALUE 'DATAB'.
01 DATAC       PIC X(200) VALUE 'DATAC'.
01 DATAD       PIC X(200) VALUE 'DATAD'.
01 DATAE       PIC X(200) VALUE 'DATAE'.
01 DATAF       PIC X(200) VALUE 'DATAF'.
01 DATAG       PIC X(200) VALUE 'DATAG'.
PROCEDURE DIVISION.
MAIN SECTION.
START000.
***** 5 CALLS TO IDISNAP
CALL "IDISNAP".
MOVE "0000" TO PARM1.
CALL "IDISNAP" USING PARM1.
MOVE "0001" TO PARM1.
CALL "IDISNAP" USING PARM1 PARM2.
MOVE "0002" TO PARM1.
CALL "IDISNAP" USING PARM1 PARM2 PARM3.
MOVE "000V" TO PARM1.
CALL "IDISNAP" USING PARM1 PARM2 PARM3 PARM1 PARM2WORK
DATAA DATAB DATAC DATAD.
GOBACK.
END PROGRAM COBMST4X.
```

### Example 2 (PL/I)

The following is an example of a PL/I program calling IDISNAP four times, showing several of the different invocation styles.

```
*PROCESS COMPILE,ATTRIBUTES,AGGREGATE,MAP,LIST,ESD,NEST;
@960IDI:PROC OPTIONS(MAIN) REORDER;
DCL WKPTR          PTR ;
DCL WORK           CHAR(4) INIT('0001') ;
DCL WORK140         CHAR(140) INIT(' ');
DCL WORK1402        CHAR(140) INIT(' ');
DCL NUMWK          FIXED DEC(9) INIT(0) ;
DCL NUMWK2         FIXED DEC(9) INIT(0) ;
DCL PICWK          PIC'999' INIT(0);
DCL IDISNAP EXTERNAL ENTRY;
/* ON ERROR CALL PLIDUMP(' F B ') */
/* ON ERROR CALL IDISNAP(WORK,WORK140) */
FETCH IDISNAP;
CALL SUBA;
```

## Using the program SNAP interface (IDISNAP)

```
SUBA: PROCEDURE ;
      CALL SUBB;
END SUBA;
SUBB: PROCEDURE ;
/* THIS WILL CALL IDISNAP 4 TIMES THEN ABEND FOR CALL 5 */
CALL IDISNAP;
DISPLAY ('ZZZ RETURNED FROM IDISNAP TO SUBB');
CALL IDISNAP('0000');
DISPLAY ('ZZZ RETURNED FROM IDISNAP(0000) TO SUBB');
WORK140 = 'USER TITLE DATA.';
CALL IDISNAP(WORK,WORK140);
WORK = '0002';
WORK140 = 'USER TITLE DATA.';
WORK1402 = 'DATASETS(IDIHIST(NULLFILE))';
CALL IDISNAP(WORK,WORK140,WORK1402);
PICWK = NUMWK2 ;
PICWK = NUMWK2 ;
PICWK = NUMWK2/NUMWK ;
END SUBB;
END @960IDI;
```

### Example 3 (Assembler)

The following is an example of an assembler program calling IDISNAP.

```
TITLE 'HLASM EXAMPLE'
R0    EQU  0
R1    EQU  1
R3    EQU  3
R13   EQU  13
R14   EQU  14
R15   EQU  15
ASMSNAP CSECT
ASMSNAP AMODE 31
ASMSNAP RMODE ANY
PRINT  GEN
STM   14,12,12(R13)
LR    R3,R15
USING ASMSNAP,R3
LA    R1,REGSAVE
ST    R13,4(,R1)
LR    R13,R1
WTO   'START OF ASMSNAP'
LOAD  EP=IDISNAP
LTR   R15,R15
BNZ   ERROR
LR    R15,R0
LA    R1,0
CALL  (15)           CALL IDISNAP
WTO   'END OF ASMSNAP'
SR    R15,R15          RC=0
B     RETURN
ERROR WTO   'ERROR LOADING IDISNAP'
RETURN L    R13,4(,R13)
L    14,12(,R13)
LM    R0,12,20(R13)
BR    R14              RETURN TO CALLER
DROP  ,
REGSAVE DS   18F
LTORG
END   ASMSNAP
```

## Invoking Fault Analyzer from Java catch block

To capture the current Java state from a Java program, Fault Analyzer can be invoked to create a history file fault entry, along with an associated MVS SVC dump. The Fault Analyzer history file that is used is the default history file, or if a different history file is desired, one can be specified via the \_IDI\_OPTS or \_IDI\_OPTSFILE environment variables described in Chapter 33, “Options,” on page 451. The fault entry that is created can subsequently be reanalyzed using the Fault Analyzer ISPF interface to review the Java and native code which was executing when Fault Analyzer was called.

In order to facilitate Java dump capture, the following is required:

- ALTER access must be granted to the Fault Analyzer IDI\_SDUMP\_ACCESS XFACILIT profile.

Set up an XFACILIT class profile with the name IDI\_SDUMP\_ACCESS (which is the same profile used for recovery fault recording SDUMP access) and provide ALTER access to the user IDs or groups for which Java dump capture is required. The following define would permit Fault Analyzer to create Java capture SDUMPS for all users in the JDEV group, if their Java application encounters an exception.

```
RDEF XFACILIT IDI_SDUMP_ACCESS UACC(NONE)
PERMIT IDI_SDUMP_ACCESS CLASS(XFACILIT) ID(JDEV) ACCESS(ALTER)
```

The ALTER access is to the XFACILIT IDI\_SDUMP\_ACCESS profile, it is not to the actual SDUMP data sets. Fault Analyzer uses authorized state to permit access to Java capture SDUMPs. The IDI\_SDUMP\_ACCESS profile acts as a switch Fault Analyzer can check to see if SDUMPs can be created for a given user.

- The MVS post-dump exit IDIXTSEL must be installed. For details, see “Installing the MVS post-dump exit IDIXTSEL” on page 313.
- The IDIS subsystem must be started. For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239.

The call to Fault Analyzer can be placed within a catch block, or anywhere else in your program, and is performed using the com.ibm.faultanalyzer.Snap.Dump method.

### Syntax

```
►—com.ibm.faultanalyzer.Snap.Dump—"—comment—"►—►
```

An optional *comment* character string can be specified, which is used to initialize the Fault Analyzer fault entry user title field.

The following is an example showing how Fault Analyzer might be called from within a Java catch block:

```
public class JavaTest {
    public static void main() {
        ...
        try {
            ...
        } catch() {
```

## Invoking Fault Analyzer from Java catch block

```
    ...
    com.ibm.faultanalyzer.Snap.Dump("Java error"); // Call Fault Analyzer
}
}
```

To facilitate calling the com.ibm.faultanalyzer.Snap.Dump method, the Fault Analyzer provided IDIXJAVA Java library must either exist in the application program build path, or be available via the current ClassPath.

The IDIXJAVA Java library can be obtained through the following steps:

1. Perform binary FTP transfer of IDI.SIDIDOC1(IDIXJAVA) to your project development directory as file IDIXJAVA.

**Note:** The IDI.SIDIDOC1(IDIXJAVA) data set and member was created as part of the Fault Analyzer SMP/E installation and might exist with a different high-level qualifier.

2. Do one of the following:
  - a. Configure your project build path to include IDIXJAVA as an external JAR library dependency.
  - b. Add the directory and file name to the current ClassPath.

---

## Invoking Fault Analyzer from Java dump events

The Java -Xdump environment switch can be used to take a TDUMP when an exception occurs and pass that TDUMP on to Fault Analyzer to generate an analysis report. For general detail about using the -Xdump settings go to [http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.60/html/dump\\_agents.html](http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.60/html/dump_agents.html) This reference describes the Java internal events which can trigger a dump and the filters which can be applied.

The following setup is an example of using -Xdump to get a Fault Analyzer report for a NullPointerException or a SocketException:

---

```
java -Xdump:system:events=throw+catch+uncaught,filter=*&NullPointerException*,opts=IEATDUMP
-Xdump:system:events=throw+catch+uncaught,filter=*&SocketException*,opts=IEATDUMP
-Xdump:tool:events=throw+catch+uncaught,filter=*&NullPointerException*,
exec="tso 'idida dsn(%last) datasets(idihist(da.test.gui.hist))' "
-Xdump:tool:events=throw+catch+uncaught,filter=*&SocketException*,
exec="tso 'idida dsn(%last) datasets(idihist(da.test.gui.hist))' " Java-program-name
```

---

The 'system' dump agent with opts=IEATDUMP is used to capture a raw TDUMP, and then the 'tool' dump agent for the same event using the exec= option calls Fault Analyzer, passing the TDUMP via the dsn(%last) token.

The "exec" in the Xdump:tool options provides Java with a command to invoke when the requested events are triggered. Because the "exec" string needs to be double quoted and the resulting -Xdump strings are long, it can be easier if the entire sequence is provided through the STDPARM DD in the BPXBATCH job that is used to run the Java program.

The range suboption can be used to limit the number of dumps generated for a specific exception. In the above example, providing a range 1..4 means that dumps are only generated for four NullPointerException or SocketException throw events,

and any more are ignored. This option is especially useful in cases where exceptions are caught and rethrown, but it does mean future Exceptions are not processed.

The priority suboption can be used to make sure dumps are created in a specific order. So, if a Java dump or Heap dump is also required, then to prevent Fault Analyzer being called on those dumps, make sure the non-TDUMPs either have lower or higher priority than BOTH the Xdump system and Xdump tool together.

---

## Dump registration processing

Unlike the SYSABEND, SYSMDUMP, and SYSUDUMP processes, which run in the user address space, the SVC dump process in MVS runs from the DUMPSRV address space. This difference means that the MVS change options/suppress dump exit, which is one of the normal means of invoking Fault Analyzer, does not work for SVC dumps. For SVC dumps, Fault Analyzer provides the IDIXTSEL exit module. SVC dumps occur for system abends, and are also used by CICS for its system dumps.

If the IEAVTSEL post-dump exit, IDIXTSEL, is installed (see “Installing the MVS post-dump exit IDIXTSEL” on page 313), then a “skeleton” fault entry is created whenever an SVC dump is written. This processing differs from normal real-time processing in that no analysis is performed, and therefore no report or minidump is produced. This Fault Analyzer process is known as “dump registration”.

The dump registration processing permits the use of two user exits which effectively are the equivalent of the normal Analysis Control and Notification user exits. These are specified using the DumpRegistrationExits option (see “DumpRegistrationExits” on page 466).

The dump registration fault entry contains only limited information, such as the time of its creation, the system name, and the name of the job that caused the SVC dump to be written. If available, the abend code and abending program name is also provided. However, the first reanalysis of the dump registration fault entry refreshes the fault entry and save a report and minidump with it—for details, see “Refresh processing” on page 171.

---

## Real-time exclusion processing

While real-time analysis is never performed if the primary subsystem (JES) is unavailable, there are a number of ways to selectively exclude various elements of the Fault Analyzer processing, as illustrated in Figure 5 on page 28.

## Real-time exclusion processing

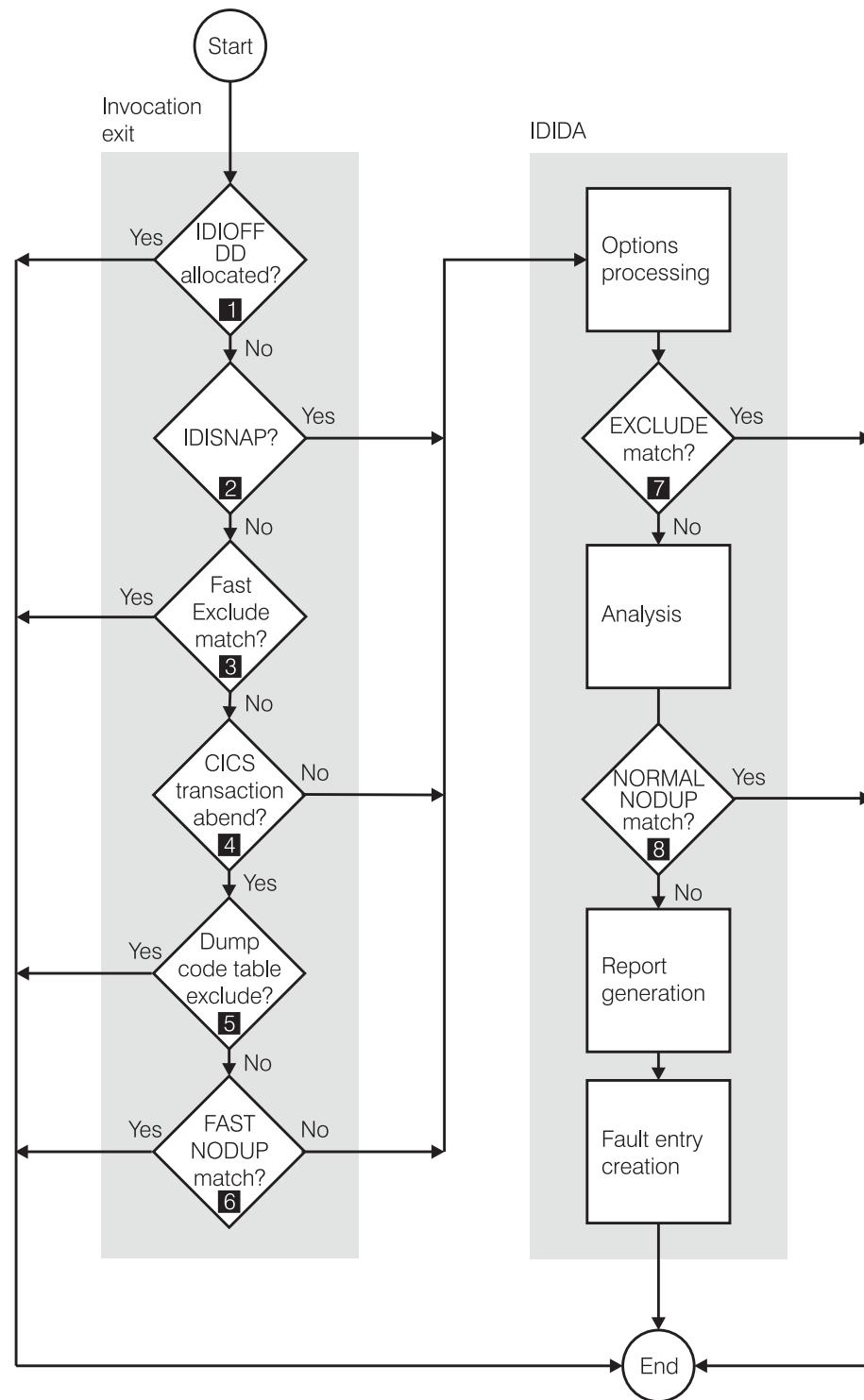


Figure 5. Real-time exclusion processing overview

Notes relating to Figure 5:

- 1** By providing an allocation of DDname IDIOFF to the abending job step, Fault Analyzer processing is immediately terminated without producing an analysis report or writing a history file fault entry. You can do this, for example, adding the following JCL statement in your JCL:

```
//IDIOFF DD DUMMY
```

Allocating IDIOFF is the quickest way to prevent Fault Analyzer from running for a particular job step, and the one which is recognized with the least amount of overhead.

**Note:** In the z/OS Unix System Services environment, setting the environment variable \_IDI\_OFF to "Y" is equivalent to using the IDIOFF DDname switch. For more information, see "Turning off Fault Analyzer using an environment variable (\_IDI\_OFF)" on page 357.

- 2 If calling IDISNAP from within your application (see "Using the program SNAP interface (IDISNAP)" on page 20), then no further exclusions are available prior to options processing being performed.
- 3 If fast Exclude options processing is enabled, and the job is eligible (see "Fast Exclude options processing" on page 282), then a matching Exclude option (see "Exclude/Include" on page 469) can be used to terminate Fault Analyzer processing early.
- 4 Additional invocation exit exclusions are provided for CICS transaction abends only.
- 5 If the CICSDumpTableExclude option is in effect (see "CICSDumpTableExclude" on page 456), and the CICS transaction abend that is associated with the fault is specified in the CICS transaction dump code table to not require a CICS dump, then no further processing is performed. That is, no analysis report is produced and no fault entry is written.
- 6 If either of the following is true, then processing terminates for the current fault:
  - The NODUP(CICSFAST(...)) option specifies a non-zero number of minutes, and the criteria used for determination of duplicate CICS transaction abend fault entries match.
  - The NODUP(IMAGEFAST(...)) option specifies a non-zero number of minutes, and the criteria used for determination of duplicate IMS transaction abend fault entries match.

Although no analysis report is produced and no history file fault entry is written, the duplicate count is still updated in the history file cache against the fault entry when the next non-duplicate fault entry is written.

For details about these fast duplicate detection options, see "NoDup" on page 482.

- 7 The EXCLUDE option (see "Exclude/Include" on page 469) can be used to terminate Fault Analyzer processing once the options have been read in the mainline code.
- 8 Prior to writing the history file fault entry, the NODUP(NORMAL(...)) option is checked. If the option specifies a non-zero number of hours and the criteria used for determination of duplicate fault entries match (see "NoDup" on page 482), then processing terminates for the current fault. The real-time report is written to IDIREPRT, but no history file fault entry is created.

Note that the NODUP(NORMAL(...)) option applies to all fault entries, including CICS transaction abend faults.

Unless Fault Analyzer processing is excluded using the IDIOFF DDname switch, then an SMF type 89 record is written to indicate Fault Analyzer usage.

## Duplicate fault processing

### Duplicate fault processing

Fault Analyzer provides two different types of duplicate fault processing, "fast" and "normal", where "fast" implies pre-analysis detection and "normal" implies post-analysis detection.

The "fast" duplicate fault type is further divided into two different subtypes, one at a CICS region level and one that encompasses an entire MVS image. The latter is applicable to IMS only.

A fault that is not deemed a "fast" duplicate is still subject to "normal" duplicate processing.

The different types of duplicate processing are controlled using the NoDup option (for details, see "NoDup" on page 482). While the NoDup option description provides a detailed explanation of each type, the following is provided as a general overview for easier comparison.

Table 3. Duplicate processing type comparison

Aspect of processing	ImageFast	CICSfast	Normal
Controlled using option	NoDup(ImageFast(IMS(...)))	NoDup(CICSfast(...))	NoDup(Normal(...))
Applicable to	All faults that use IMS, except CICS transaction faults	Only CICS transaction faults	All faults
Order of processing	1	1	2
Improves performance by suppressing fault analysis (that is, no IDIDA TCB attach)	Yes	Yes	No
Saves disk space by suppressing history file fault entry	Yes	Yes	Yes
Requires IDIS subsystem started	Yes	No	No
Duplicate signature repository location	IDIS subsystem storage	CICS region storage	History file index
Default setting	Enabled, 5 minutes	Enabled, 5 minutes	Enabled, 24 hours

The number of duplicate faults that have occurred against a given history file fault entry is shown in the Fault Entry List display DUPS column (for details, see "The Fault Entry List display" on page 34).

## Recovery fault recording

The recovery fault recording feature of Fault Analyzer is provided to reduce the number of instances where an abnormal termination problem during real-time analysis prevents a normal fault entry from being created. This approach might, for example, be in the following situations:

- Insufficient storage. (See "IDI0005S" on page 535.)
- Fault Analyzer abended. (See "IDI0047S" on page 540.)
- Fault Analyzer timed out. (See "IDI0092S" on page 546.)
- Invalid negative storage length request. (See "IDI0105S" on page 547.)

When a terminating condition is subject to recovery fault recording processing, then a skeleton fault entry is created and an associated SDUMP (SVC dump) or IEATDUMP (transaction dump) written.

First a check is made to see if security access is granted to use SDUMP as the recovery fault recording dump type, since this type is the preferred dump type for performance reasons. (For details, see "Using the XFACILIT resource class for SDUMP RFR data sets" on page 235.)

**Note:** The IDIXTSEL SVC dump registration exit is required to support the recovery fault recording feature when using SDUMPs. For details of this exit, see "SVC dump registration" on page 227.

If SDUMP cannot be used, then IEATDUMP is instead used as the recovery fault recording dump type.

The term "RFR dump" is used to refer to the recovery fault recording dump data set, regardless of which dump type is used.

The RFR dump creates an extra data set, into which MVS writes a dump of the address space. This data set takes significantly more DASD space than a minidump, but in these situations, Fault Analyzer has failed to gather the minidump. Subsequently, the RFR dump data set is used in place of the minidump for reanalysis of the skeleton fault entry.

**Note:** To enable recovery fault recording processing, the IDIS subsystem must be started.

The history file in which the fault entry is created is either the current history file for the abending job, as determined at the time of the abnormal analysis termination, or the dehistory file for the IDIS subsystem. The current history file that is determined for the abending job is attempted to be used first if it is a PDSE. Otherwise, the IDIS subsystem dehistory file is used.

Message IDI0126I is issued to indicate in which history file the fault entry was created.

If the RFR dump is an IEATDUMP, then it is created from the abending region. However, if it is an SDUMP, then it is created by the IDIS subsystem. The skeleton fault entry is always created by the IDIS subsystem.

Once the recovery fault recording process starts, no user exits are driven for the process, except for any Notification user exits specified in the options available to the IDIS subsystem, which are invoked when creating the skeleton recovery fault recording fault entry. To distinguish a recovery fault recording event from other invocations of Notification user exits, the NFY.NFYTYPE field is set to 'R'. For details about the Notification user exit, see "Notification user exit" on page 397.

If the RFR dump is an IEATDUMP, then the name of the IEATDUMP data set created is controlled by the name in the IDIRFRDS CSECT. For details about the use of this name, and information about how to change it, see "Changing the default recovery fault recording IEATDUMP data set name" on page 258. To permit automatic deletion of IEATDUMP data sets when the fault entries they are associated with are deleted, then changing the default high-level qualifier might be required, subject to installation-specific security rules. See "Managing recovery fault recording data set access" on page 234 for more information.

## Recovery fault recording

Depending on where in the real-time analysis process the problem occurred, reanalysis of the recovery fault recording fault entry is capable of producing a reanalysis report, which is effectively identical to the one that would have been produced if the real-time analysis had completed normally. The fact that a recovery fault recording fault entry was created instead of the normal real-time fault entry is almost transparent to the user for many of the recovery situations.

When a recovery fault recording fault entry is deleted, then the associated RFR dump data set is also automatically deleted to ensure that these data sets are not taking up disk space unnecessarily. Failure to delete the RFR dump data set results in message IDI0128I to be issued. See "IDI0128I" on page 551 for the conditions under which the recovery fault recording dump data set is deleted.

## RFR dump titles

The Fault Analyzer recovery fault recording dump title depends on whether the dump is an IEATDUMP or an SVCDUMP.

### IEATDUMP dump title

►►—*history-file-name(fault-id)*^—*dump-data-set-name*^—————►►

**Note:** In the above, ^ represents the non-printable character X'00'.

The following is an example of a Fault Analyzer IEATDUMP recovery fault recording dump title:

MY.HIST(F32752).

IDIRFRHQ.IDIRFR.FAE1.D110216.T003630.IDIVPCO.

### IEATDUMP dump title

►►—*history-file-name(fault-id)*^—^SVCDUMP(0xasid)^—————►►

**Note:** In the above, ^ represents the non-printable character X'00'.

The following is an example of a Fault Analyzer SVCDUMP recovery fault recording dump title:

MY.HIST(F32753).

.SVCDUMP(0x0080).

---

## Using SLIP,COMP=0C4 with Fault Analyzer

Like many other products, Fault Analyzer employs ESTAE processing to protect and recover from S0C4 abends, where these can be expected to occur during normal processing. If a SLIP trap is set to capture S0C4 abends on your system, then it is likely that unwanted matches occur as a result of these. To prevent such unwanted matches, qualify the SLIP trap by using other parameters, such as DATA and PVTMOD, or add an extra SLIP trap as follows:

SLIP SET, ID=xxxx, COMP=0C4, ACTION=IGNORE, *location*=IDIDA, END

where *location* is LPAMOD if the IDIDA load module is placed in LPA, otherwise PVTMOD.

---

## Chapter 3. The Fault Analyzer ISPF interface

At any time after an abend you can, as a TSO user, start the Fault Analyzer ISPF interface to review the fault. Using this interface you can:

- View the stored real-time analysis report.
- Start a batch reanalysis (for details, see Chapter 4, “Performing batch reanalysis,” on page 95).
- Start an interactive reanalysis (for details, see Chapter 5, “Performing interactive reanalysis,” on page 103).
- View information about the fault.
- View details about any faults that might have occurred, that were deemed to be duplicates of the current fault.
- Delete the fault entry.

The ISPF interface also permits you to:

- Analyze CICS system abend dumps (for details, see Chapter 6, “Performing CICS system abend dump analysis,” on page 177).
- Analyze Java dumps (for details, see Chapter 8, “Performing Java analysis,” on page 189).

**Note:** Whereas the information in this chapter assumes that the Fault Analyzer ISPF interface is invoked under ISPF, it is possible to instead invoke this interface under CICS. When doing so, restrictions might apply. These restrictions are described in “Performing interactive reanalysis under CICS” on page 206.

**Reanalysis:** You can only perform reanalysis of a fault if either a minidump, or a SYSMDUMP or SVC dump, was captured at the time of the abend.

Compiler listing or side file data sets that were allocated or specified via the DataSets option when the real-time analysis took place, are reused if performing reanalysis (if they are available in the reanalysis environment).

To make the reanalysis different from the initial real-time analysis, you must do one (or more) of the following:

- Supply compiler listings (or side files) for the programs involved in the abend (if they were not available for the initial real-time analysis).
- Change analysis options.
- Use the interactive reanalysis to review dump storage.

The main differences between the batch and interactive reanalysis steps are:

- Interactive reanalysis always provides full detail, and lets you look at storage locations that might not be included in the analysis report, whereas batch reanalysis provides the level of detail you ask for through the Detail option, and does not let you look at storage locations.
- Interactive reanalysis ties up the use of an ISPF session, whereas once you submit batch reanalysis jobs you can get on with other things.

If you want to supply a listing or side file so that Fault Analyzer can provide source line information when it performs the fault reanalysis, you must compile

## The Fault Analyzer ISPF interface

the program and then store the compiler listing or side file. For more information on this process, see Chapter 20, "Providing compiler listings or Fault Analyzer side files," on page 285.

If you have already created the listing or side file and are holding it in a non-standard storage location, you can use JCL DD statements to point to the location. "Pointing to listings with JCL DD statements" on page 18 sets out possible values.

---

## Invoking the interface

How you invoke the Fault Analyzer ISPF interface depends on how it was customized at your site, but it is generally done by choosing an option from an ISPF selection panel, or by issuing a command.

Various suggested ways of how to invoke Fault Analyzer are described in Chapter 15, "Modifying your ISPF environment," on page 247.

If you do not know how to invoke the Fault Analyzer ISPF interface at your site, then ask your systems programmer or the person who customized Fault Analyzer.

To display the on-line help while in the interface, press the Help function key (PF1).

## ISPF split screen support

Multiple concurrent invocations of Fault Analyzer by a single TSO/ISPF user (for example, using ISPF split screens) is supported, but with some limitations. For example:

- If performing concurrent interactive reanalysis of the same fault entry in multiple ISPF split screen sessions, changes to user notes in one session are not reflected in another, and only user notes from the analysis that is ended last are saved.
- The last used history file and fault entry lists available from the "File" pull-down menu of the Fault Entry List display are maintained for each ISPF session separately. The last Fault Analyzer ISPF interface session, during which a change was made to this information, that is ended, updates the information in the user's ISPF profile.

---

## The Fault Entry List display

The Fault Entry List display is shown when the Fault Analyzer ISPF interface is started. Figure 6 on page 35 shows an example of a Fault Entry List display:

## The Fault Entry List display

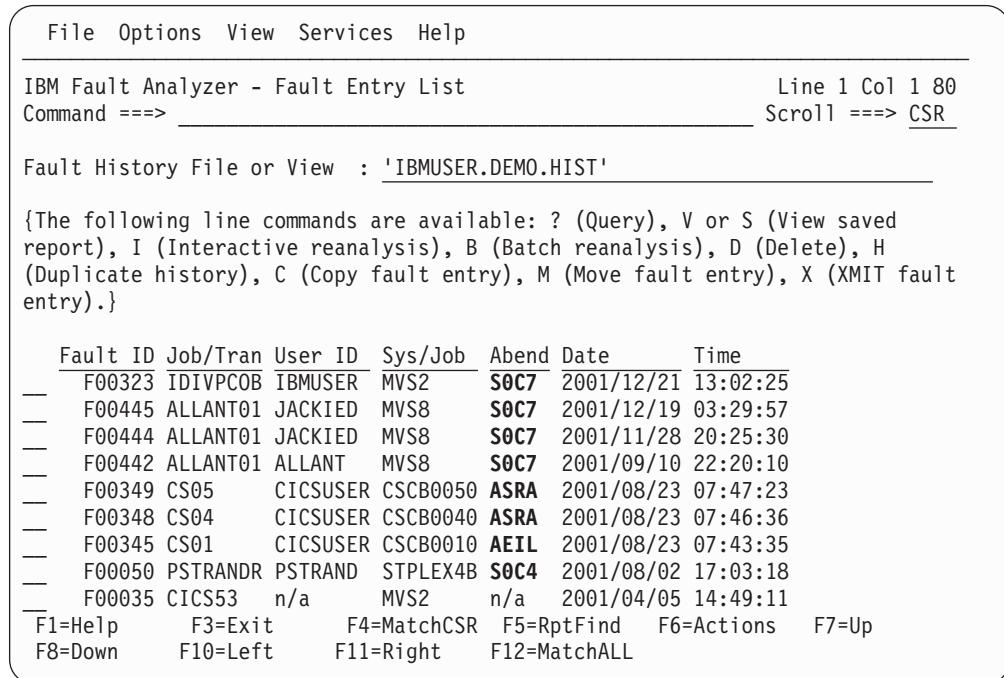


Figure 6. Sample Fault Entry List display

**Note:** If your Fault Entry List display does not show the PF keys, and you would like to see them, then enter the ISPF command:

FKA ON

Fields shown in yellow (default color) are point-and-shoot enabled. This enablement means that you can place the cursor on such fields and press the Enter key to display more information. For example, by selecting an abend code in the Abend or I\_Abend columns, the explanation for that abend code is displayed.

The history file or view (see “Using views” on page 37) that was last selected while using the Fault Analyzer ISPF interface is shown by default. The first time the interface is used, the initial history file name is obtained using the IDIHIST suboption of the DataSets option in effect. To select another history file or view, refer to “Changing the history file or view displayed” on page 37.

If a view was selected the last time the ISPF interface was used, and the view contains errors, then it is possible that an error display is presented prior to the Fault Entry List display. An example of an error display is shown in Figure 7 on page 36.

## The Fault Entry List display

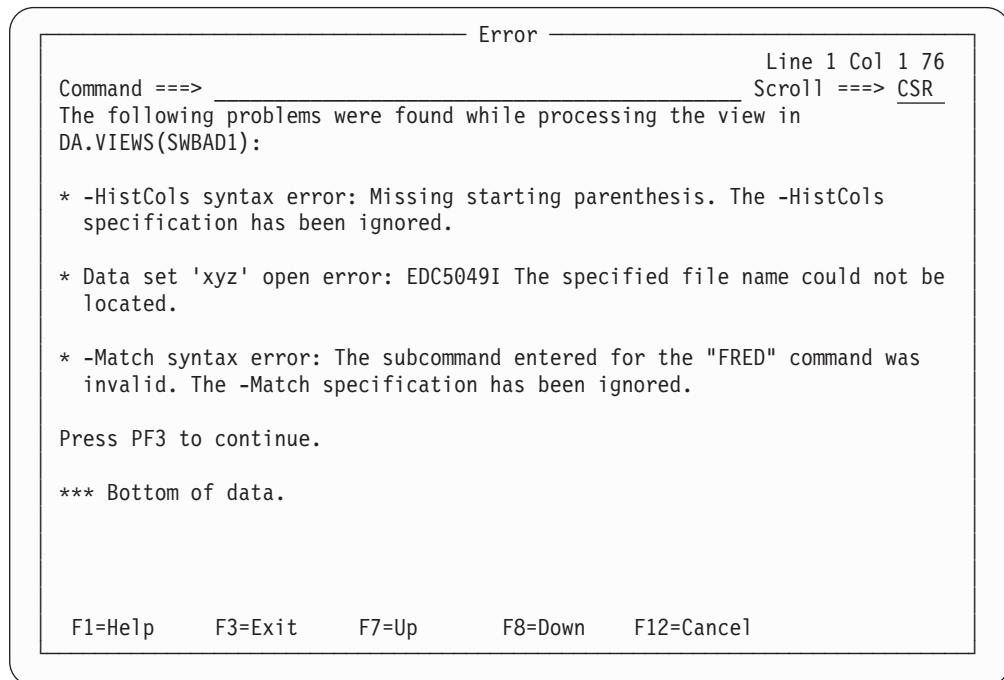


Figure 7. Sample Error display

To exit from the error display, press PF3.

The error display is shown each time the incorrect view member is read, and might therefore also be shown when, for example, the Fault Entry List Column Configuration display is presented. The identified errors in the view should be corrected to avoid this display.

Entries in the Fault Entry List display are by default listed in reverse chronological order with the most recent fault entry (based on abend date and time) shown at the top.

Each fault entry in the list occupies a single line and is identified by a fault ID on the left side of the display. Other information that might be displayed for each fault entry can be determined by the user—for details on this, refer to “Fault entry list column configuration” on page 43. The default information displayed, when no HistCols option is specified and no customization is made by the user, is as shown in Figure 6 on page 35.

You can use the displayed fields to identify the faults you are interested in, or reduce the display to only a subset of the faults—for details on how to do this, refer to “Sorting and matching fault entries” on page 47.

As shown at the top of the display if help text is enabled (as illustrated in Figure 6 on page 35), a number of line commands are available against individual history file entries. For details on these, see “Applying an action to a particular fault” on page 52. For information about how to show or hide help text, see “Adding or removing help text” on page 76.

This screen responds to the standard UP, DOWN, LEFT, and RIGHT commands, which by default are assigned to the PF7, PF8, PF10, and PF11 function keys respectively. These can be used to scroll the display horizontally or vertically as needed to see all of the information available.

Optional help text that lists the available line commands is displayed only when the top-most line of the display is shown. If the display is scrolled down any number of lines, this help text disappears, but reappears again if the display is scrolled to the top. For general information about help text, see “Adding or removing help text” on page 76.

The history file or view input field, and the column headings, are never scrolled out of view. However, if scrolling horizontally, the column headings scroll with the data below them.

The line command input fields on the left side of the display remains in that position regardless of any horizontal scrolling.

In the top right corner of the screen is the current top-most line number and indication of the left-most and right-most columns currently displayed.

The end of the fault entry list is indicated by the line:

\*\*\* Bottom of data.

This line is used to indicate the bottom of all Fault Analyzer ISPF interface scrollable displays.

You exit from the Fault Analyzer ISPF interface by issuing the Exit command (PF3) from the Fault Entry List display, or by selecting the Exit Fault Analyzer option from the Fault Entry List display File menu.

## Using views

When it would be useful to concurrently view history file entries from more than a single history file:

- A view name can be specified instead of a history file name on the Fault Entry List display.
- A view name can be selected via the File menu List Views option. For information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61.

The definition of these views must be set up in the PDS(E) data set identified by the IDIVIEWS suboption of the DataSets option in the IDICNF00 parmlib config member.

With views you can:

- View fault entries from multiple history files simultaneously.
- Provide a specific column layout for the Fault Entry List display (see “Specifying a default column layout” on page 264).
- Provide a selection criteria for the initially displayed list of fault entries (see “Specifying an initial fault entry selection criteria” on page 264).

For information about how to set up views, see “Setting up views” on page 263.

## Changing the history file or view displayed

When the Fault Analyzer ISPF interface is started initially, the history file or view last displayed is shown. To select a different history file or view, do one of the following:

### Type a different history file or view name

To specify a history file or view name to be displayed, type its name on the “Fault History File or View” line as the example shown at **1** in

## The Fault Entry List display

Figure 8 where the history file name 'MY.HIST' is selected. After typing the history file or view name, press the Enter key to show the fault entries.

The screenshot shows the 'IBM Fault Analyzer - Fault Entry List' window. At the top, there is a menu bar with 'File', 'Options', 'View', 'Services', and 'Help'. Below the menu is a toolbar with buttons for 'Line 1 Col 1 80', 'Scroll ==> CSR', and 'Fault History File or View : my.hist [1]'. A message box displays: '{The following line commands are available: ? (Query), V or S (View saved report), I (Interactive reanalysis), B (Batch reanalysis), D (Delete), H (Duplicate history), C (Copy fault entry), M (Move fault entry), X (XMIT fault entry).}' Below this, a table lists fault entries:

Fault ID	Job/Tran	User ID	Sys/Job	Abend	Date	Time
F00323	IDIVPCOB	IBMUSER	MVS2	SOC7	2001/12/21	13:02:25
F00445	ALLANT01	JACKIED	MVS8	SOC7	2001/12/19	03:29:57
F00444	ALLANT01	JACKIED	MVS8	SOC7	2001/11/28	20:25:30
F00442	ALLANT01	ALLANT	MVS8	SOC7	2001/09/10	22:20:10
F00349	CS05	CICSUSER	CSCB0050	ASRA	2001/08/23	07:47:23
F00348	CS04	CICSUSER	CSCB0040	ASRA	2001/08/23	07:46:36
F00345	CS01	CICSUSER	CSCB0010	AEIL	2001/08/23	07:43:35
F00050	PSTRANDR	PSTRAND	STPLEX4B	SOC4	2001/08/02	17:03:18
F00035	CICS53	n/a	MVS2	n/a	2001/04/05	14:49:11

At the bottom of the window, there is a status bar with keyboard shortcuts: F1=Help, F3=Exit, F4=MatchCSR, F5=RptFind, F6=Actions, F7=Up, F8=Down, F10=Left, F11=Right, F12=MatchALL.

Figure 8. Typing a different history file or view name

The following defines the rules for naming history files and views:

- For history file names, the standard TSO naming convention applies, that is, the name typed is automatically prefixed by the TSO prefix if not enclosed in single quotes. For example, if your TSO prefix is set to FRED and you want to specify the history file name FRED.HIST, type either

HIST

or

'FRED.HIST'

on the "Fault history file or view" line.

If missing, the ending quote is automatically added.

- View names are member names in one of the data sets that is associated with the IDIVIEWS DDname. These are specified by enclosing them in parenthesis. For example, to specify that the view member ABC is to be displayed, type

(ABC)

on the "Fault history file or view" line.

If missing, the closing parenthesis is automatically added.

To obtain a list of history files from which a selection can be made, a history file pattern can be specified using wildcards, consisting of one or more percent signs (%) and/or asterisks (\*):

- \* A single asterisk by itself indicates that at least one qualifier is needed to occupy that position. A single asterisk within a qualifier indicates that zero or more characters can occupy that position.

- \*\* A double asterisk indicates that zero or more qualifiers can occupy that position. A double asterisk within a qualifier is invalid.
- % A single percent sign indicates that any one single alphanumeric or national character can occupy that position.
- %... One to eight percent signs can be specified in each qualifier.

The following examples are valid history file patterns:

History file pattern	Resulting list
'FRED.*'	All history file names with FRED as the first qualifier and at least one more qualifier.
'FRED.**'	All history file names with FRED as the first qualifier.
'FRED.**.HIST'	All history file names with FRED as the first qualifier, HIST as the last qualifier and zero or more qualifiers in between.
'AAA%*.B*%%%B'	All history file names that start with AAA, have at least one more character in the high level qualifier, and have a second qualifier that begins and ends in B, with at least three letters between the Bs.

The rules for using quotes around history file names also apply to history file patterns.

The first qualifier of a history file pattern, after prefixing if applicable due to unquoted specification, must not consist of wildcards only, for example '\*', '\*\*', and '\*\*.HIST'. However, '\*\*.HIST' is valid if running with PREFIX ON.

### Select a previously used history file or view

A record is maintained of the last 10 history files or views displayed. To select a previously displayed history file or view, first select the File menu Last Accessed Fault History Files or Views option (for information about selecting this option in general, refer to "Action-bar pull-down menus" on page 61). This opens the Last Accessed Fault History Files or Views display as the example shown in Figure 9 on page 40.

## The Fault Entry List display

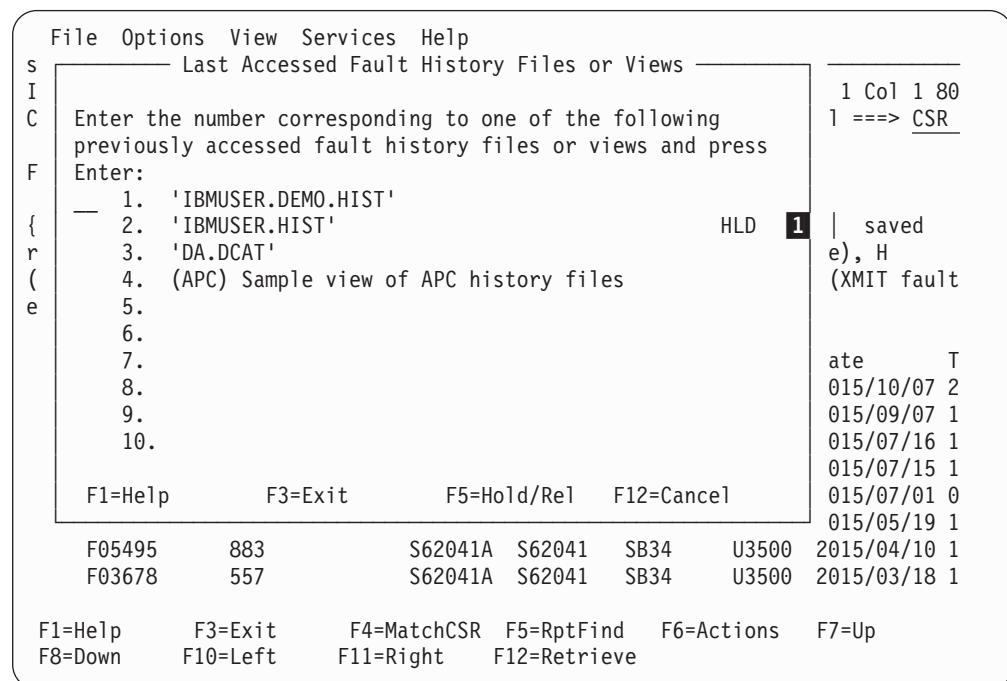


Figure 9. Sample Last Accessed Fault History Files or Views display

From the Last Accessed Fault History Files or Views display shown in Figure 9, type the number corresponding to the desired history file or view name at the initial cursor position and press Enter to display the entries for the selected history file or view.

To return to the Fault Entry List display without making any changes, press either PF3 or PF12.

To prevent a history file name from dropping off the list, enter the number of the list item in the input field and press PF5. The "HLD" indicator will be shown on the right hand side of the entry, see **1** in the above example.

If a history file name is already held ("HLD" is shown on the right hand side of the entry), then it can be released by entering the number of the list item in the input field and press PF5 again.

### Select a previously used history file entry

A record is maintained of the last 10 history file entries used. To select a previously displayed history file entry, first select the File menu Last Accessed Fault History File Entries option (for information about selecting this option in general, refer to "Action-bar pull-down menus" on page 61). This opens the Last Accessed Fault History File Entries display as the example shown in Figure 10 on page 41.

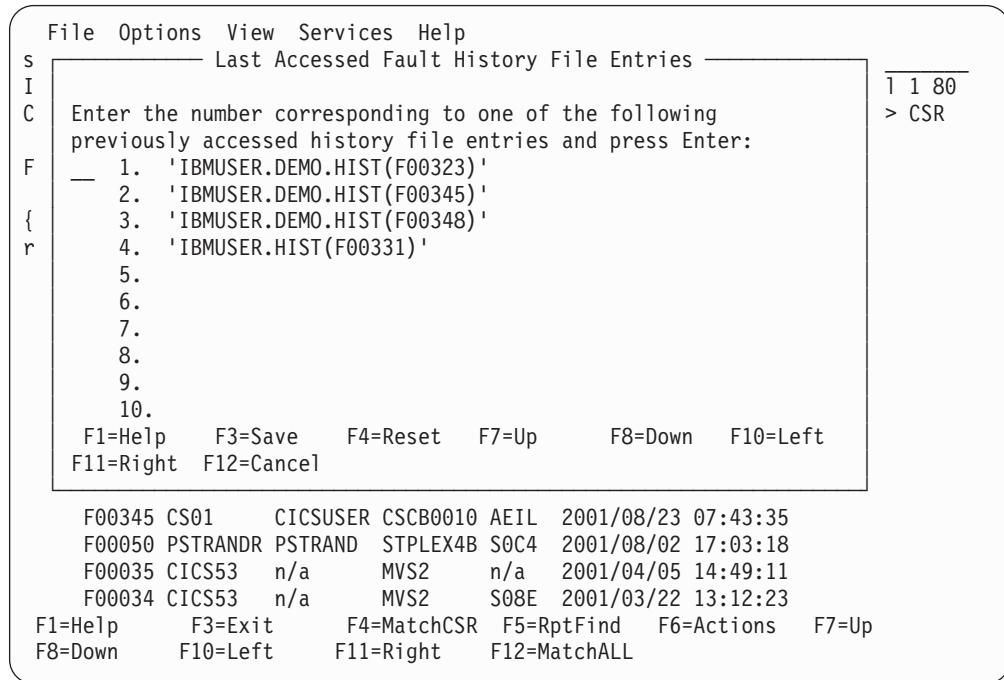


Figure 10. Sample Last Accessed Fault History File Entries display

From the Last Accessed Fault History File Entries display shown in Figure 10, type the number corresponding to the desired history file entry at the initial cursor position and press Enter to display it.

To return to the Fault Entry List display without making any changes, press either PF3 or PF12.

### Select a view from a list of views

To see a list of views available to you, select the File menu List Views option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens a display as shown in Figure 11 on page 42.

## The Fault Entry List display

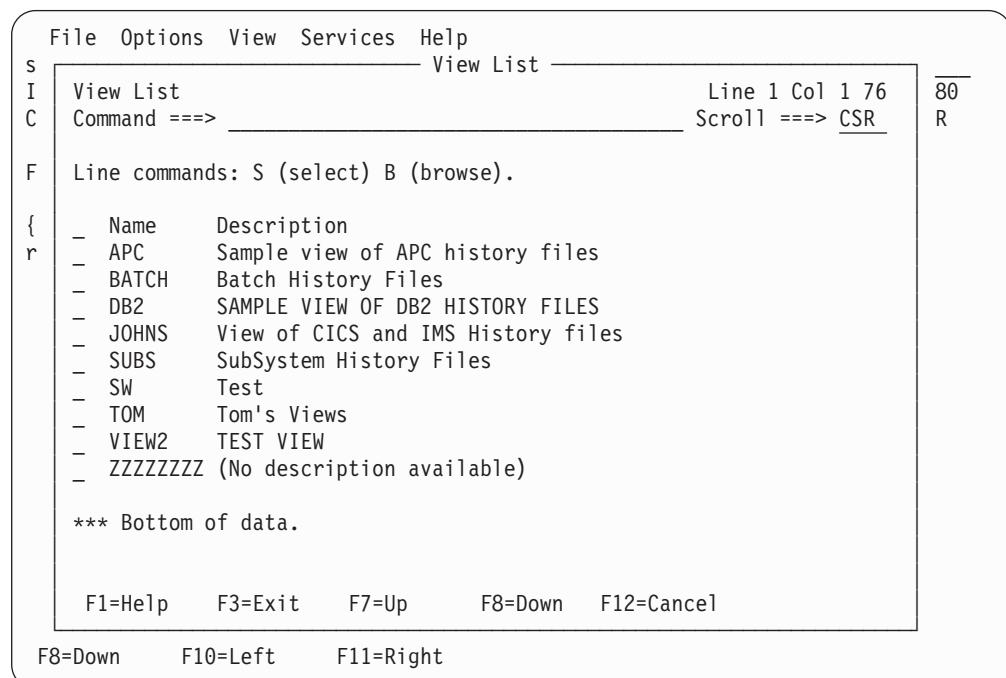


Figure 11. Sample View List display

From here, you can enter one of the following line commands against each view:

- B** This command permits you to browse the view member. For example, if the view member JOHNS was selected for browse, the contents of this member would be displayed as shown in Figure 12.

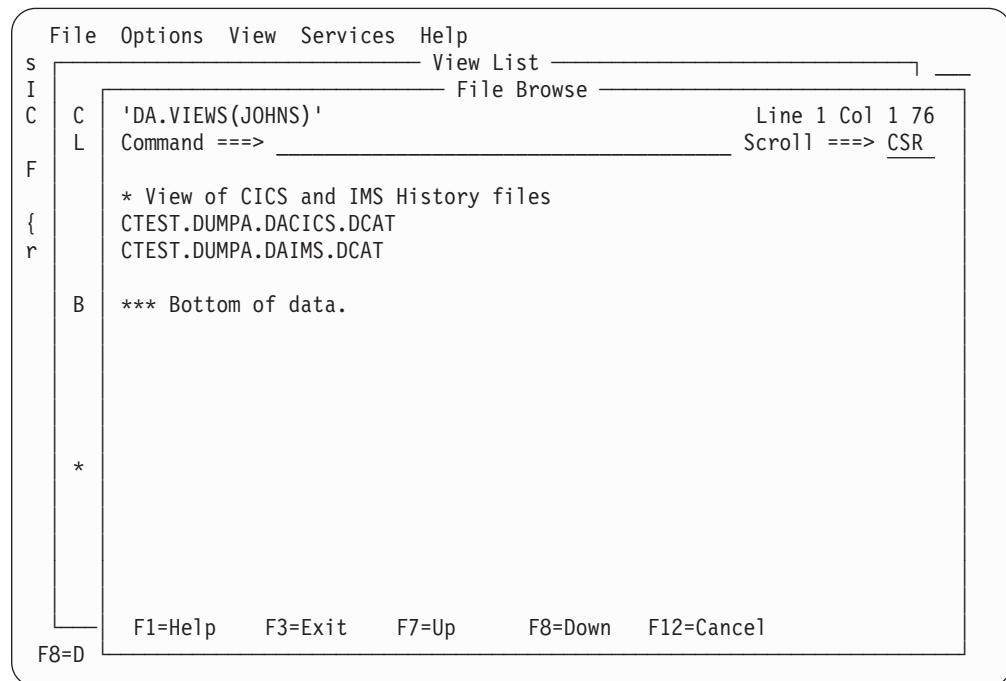


Figure 12. Sample File Browse display

To return from the File Browse display to the View List, press either PF3 or PF12.

- S** This action selects the view for display and automatically return to the previous display with the selected view name specified on the "Fault History File or View" line.

To display the chosen view, press PF3.

To return to the previously displayed history file or view without making any changes, press PF12.

When a different history file or view is selected, the column configuration of the Fault Entry List display might change.

## Fault entry list column configuration

The fault information that is shown on the Fault Entry List display is determined by the HistCols option in effect. If no HistCols option is used, the default is as illustrated in Figure 6 on page 35.

Individual users are able to alter the Fault Entry List display information by either entering the COLS command or by selecting the View menu Column Configuration option (for information about selecting this option in general, refer to "Action-bar pull-down menus" on page 61). This opens the Fault Entry List Column Configuration display as the example shown in Figure 13.

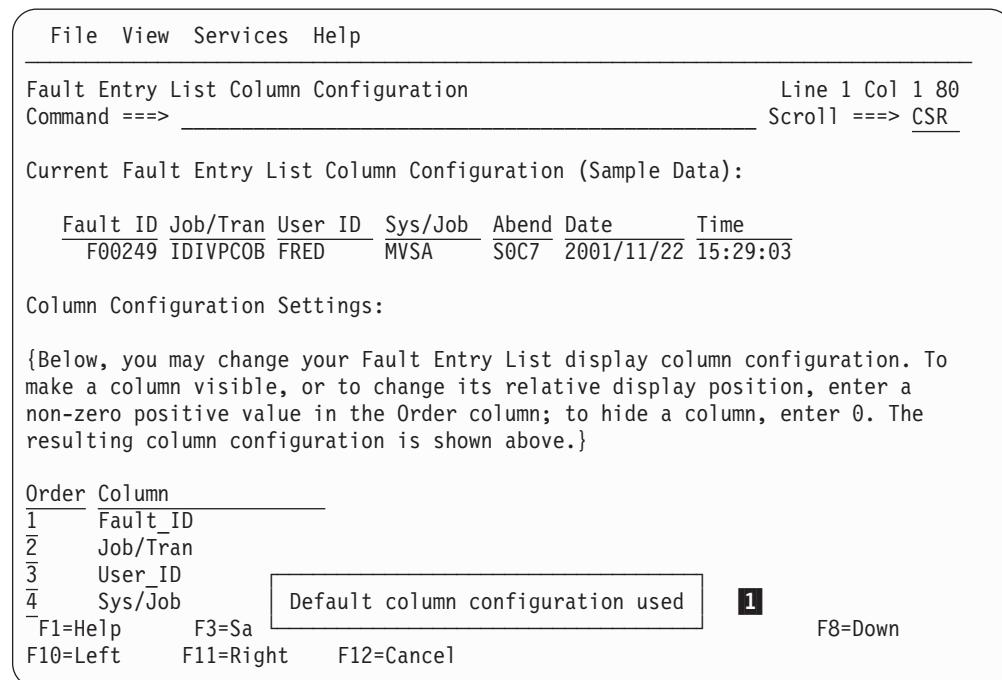


Figure 13. Sample Fault Entry List Column Configuration display

The Fault Entry List Column Configuration display is divided into two sections:

- The first is the Current Fault Entry List Column Configuration section, which shows the current column configuration with headings and sample data. This section permits you to see which of the selected columns are visible on the Fault Entry List display without first needing to scroll the display horizontally.
- The second is the Column Configuration Settings section, which permits you to modify the columns used in the Fault Entry List display.

## The Fault Entry List display

To make a column visible, or to change its relative display position, enter a non-zero positive value in the Order column; to hide a column, enter 0.

After pressing Enter, the resulting column configuration is shown in the current fault entry list column configuration section.

The Fault\_ID column cannot be hidden. If it is not given a specific display position, then it defaults to being the first column.

When the Fault Entry List Column Configuration display is first presented, a message is issued which identifies from where the current column configuration was read (see **1** in Figure 13 on page 43). There are four different possibilities:

- **Default column configuration used**

This value indicates that a history file, or a view without a valid -HistCols specification, is selected from the Fault Entry List display, and no changes to the default configuration has been saved in the user's ISPF profile. The default column configuration is determined by the HistCols option in effect.

If changes are made to this configuration, then they are saved in the user's ISPF profile as the general column configuration.

Entering the RESET command (PF4) has no effect.

- **General column configuration read from user profile**

This value indicates that a history file, or a view without a valid -HistCols specification, is selected from the Fault Entry List display, and a general column configuration exists in the user's ISPF profile.

If changes are made to this configuration, then it replaces the general configuration in the user's ISPF profile.

By entering the RESET command (PF4), the column configuration is reset to the default configuration.

- **Column configuration read from view member *member-name***

This value indicates that a view with a valid -HistCols specification is selected from the Fault Entry List display, and no changes to this configuration has been saved in the user's ISPF profile.

If changes are made to this configuration, then they are saved in the user's ISPF profile as a view-specific column configuration.

Entering the RESET command (PF4) has no effect.

- **Specific column configuration for view *member-name* read from user profile**

This value indicates that a view with a valid -HistCols specification is selected from the Fault Entry List display, and a view-specific column configuration exists in the user's ISPF profile.

If changes are made to this configuration, then it replaces the view-specific configuration in the user's ISPF profile.

By entering the RESET command (PF4), the column configuration is reset to the -HistCols specification in the view.

The SAVE command (PF3) is used to save the current column configuration in the ISPF profile and return to the Fault Entry List display with the new configuration active. All subsequent interactive Fault Analyzer sessions use this configuration until it is changed by a subsequent modification, reset to the default using the RESET command, or the ISPF profile is deleted.

The CANCEL command (PF12) can be used to return from the Fault Entry List Column Configuration display without saving any changes made.

**Available columns**

The following lists the information displayed for each of the available columns:

**Fault\_ID**

The ID assigned to the fault.

**Abend**

The initial (if more than one) abend code.

For a fault entry created using IDISNAP, the abend code is shown as "SNAP".

**Appl\_ID**

The CICS application ID.

**CICS\_Trn**

The failing CICS transaction ID. This column is only applicable to CICS.

**Class** The job class in which the job was executing.

**Date** The date when the fault occurred in LOCALE-option dependent format.

**Dups** The number of duplicate faults detected. See "NoDup" on page 482 for details about duplicate determination.

**Dup\_Count**

Deprecated—use Dups instead.

**Dup\_Date**

The date when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this date is set to the initial abend date (see Date).

**Dup\_Time**

The time when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this time is set to the initial abend time (see Time).

**EXEC\_Pgm**

The abending job step program name from the JCL EXEC PGM= parameter.

**Group\_ID**

The security server default group ID.

**History\_File\_DSN**

The history file data set name containing the displayed fault entry.

**I\_Abend**

The abend code for which Fault Analyzer was invoked.

For a fault entry created using IDISNAP, the abend code is shown as "SNAP".

**IMS\_Pgm**

The IMS application program name for faults involving IMS.

**Job/Tran**

For a CICS transaction abend, this value is the CICS transaction ID. Otherwise, it is the name of the job that abended. This column combines information from the Jobname and CICS\_Trn columns.

**Job\_ID**

The JES job ID of the abending job.

## The Fault Entry List display

### **Job\_Type**

The abending job type as one of the following:

**Batch** Batch job

**CICS** CICS transaction

### **DumpReg**

Dump registration

**STC** Started task

**TSO** TSO user

### **Jobname**

The name of the abending job.

**Lock** The fault entry lock flag. For information about the purpose of this flag, see "Viewing fault entry information" on page 82.

### **Minidump**

Indication of minidump availability as follows:

**Yes** Minidump is available

**No** Minidump is not available

### **Module**

The point-of-failure module name.

For dump registration fault entries, this value is the name of the module identified in the SVC dump header SDWA as the load module involved in the error. Following initial reanalysis and fault entry refresh, this value is updated to become the point-of-failure module name, which might be different.

### **MD\_Pages**

The number of minidump pages saved for the fault.

### **MVS\_Dump**

Indication of MVS dump availability as follows:

**Yes** MVS dump is available

**No** MVS dump is not available

**Note:** This column indicates if an MVS dump data set was available at the time of abend. The associated MVS dump data might now be deleted or not exist on the system on which the fault entry is being displayed.

### **MVS\_Dump\_DSN**

The name of any associated MVS dump data set written at the same time as when the fault occurred.

**Note:** This column indicates if an MVS dump data set was available at the time of abend. The associated MVS dump data might now be deleted or not exist on the system on which the fault entry is being displayed.

### **Netname**

CICS transaction netname.

**Offset** The point-of-failure offset.

### **Program**

The point-of-failure program name.

**Stepname**

The job step name of the abending job.

**Sys/Job**

For a CICS transaction abend, this name is the CICS job name. Otherwise, it is the ID of the system on which the job abended. This column combines information from the System and Jobname columns.

**System**

The system ID on which the abend occurred.

**Task** The failing CICS transaction task number. This column is only applicable to CICS faults.

**Term\_ID**

CICS transaction terminal ID.

**Time** The time when the fault occurred in LOCALE-option dependent format.

**Tran\_ID**

Deprecated—use CICS\_Trn instead.

**User\_ID**

For CICS and for IMS message processing regions (MPP), the user ID that is associated with the abending transaction. Otherwise, the user ID that is associated with the abending job.

**User\_Title**

User-maintained title information (see “Viewing fault entry information” on page 82 for details).

**Username**

User-maintained name information (see “Viewing fault entry information” on page 82 for details).

## Sorting and matching fault entries

The Fault Entry List display column headings are tabbable and shown in reverse highlight mode, which indicates that the table column attributes are modifiable. By placing the cursor on the heading, and pressing Enter, a Column Attributes display is presented, which allows you to sort the column data in ascending or descending order, or to show only the fault entries that satisfy a given MATCH criteria.

A sample Column Attributes display is shown in the following.

## The Fault Entry List display

File Options View Services Help							
Column Attributes		try List					
I	C	Line 28 Col 1 80 Scroll ==> CSR					
F	D	(SW) SWILKEN.HIST*					
	Column Name:	Dup_Time Jobname Abend Module System User_I					
	Dup_Date	15:57:37 PLICBPL SNAP PLICBPL FAE1 SWILKE					
	Sort:	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
	Enter "/" to select	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
	– Ascending	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
	– Descending	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
	Match:	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
	*	13:44:02 CSCB0650 SNAP CSCB0650 FAE1 TRUONG					
		12:50:07 IDIVPCOB S0C7 IDISCBCL1 FAE1 SWILKE					
		12:08:10 IDIVPCOB U4039 IDISCBCL1 FAE1 SWILKE					
		12:07:19 IDIVPCOB S0C7 n/a FAE1 SWILKE					
		12:06:36 IDIVPCOB S0C7 IDISCBCL1 FAE1 SWILKE					
		09:30:49 PLI23 U4000 @960IDI FAE1 SWILKE					
	SW16358	SW16358 2009/09/15 08:53:50 PLI23 U4000 @960IDI FAE1 SWILKE					
	SW16357	SW16357 2009/09/08 11:07:26 PLI23 U4000 @960IDI FAE1 SWILKE					
	SW16356	SW16356 2009/09/08 11:06:22 PLI23 S806 n/a FAE1 SWILKE					
	SW16355	SW16355 2009/09/08 11:05:16 PLI23 U4000 @960IDI FAE1 SWILKE					
	SW16354	SW16354 2009/09/08 10:55:24 PLIPL SNAP PLICBPL FAE1 SWILKE					

Figure 14. Sample Column Attributes display

Select the desired sort order by typing a forward slash (/) in the ascending or descending attribute input field and pressing Enter.

The MATCH attribute is case insensitive and permits the use of wildcards. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character. The default value is a single asterisk.

The value TODAY can be used in date columns to MATCH on the current date.

You can match faults, so that the display shows only the faults that share a common value for one of the fields (for example, a similar job name, or failing with the same abend code). This matching is useful when you are looking for faults with a similar pattern, or if you want to collect entries into a contiguous group so that you can apply a range delete.

Only fault entries that match the specified string are shown.

All attribute settings are cumulative, which means that once a particular sort order has been applied to a column, a sort on a different column is performed against the already sorted fault entry list. Also, once fault entries have been removed from the display due to a non-matching match criteria, the only way to restore this data is to perform a reset.

The reset can be performed by placing the cursor on the reset point-and-shoot field and pressing the Enter key. Alternatively, a RESET command can be entered on the command line.

Columns for which attributes have been changed are shown with turquoise color instead of blue.

The most recent attribute setting is shown whenever a column header is selected.

## Additional ways to match and select faults

In addition to the Column Attributes display match capability (see “Sorting and matching fault entries” on page 47), the MATCH command can be used to select only a subset of all fault entries in a history file or view.

The MATCH command is limited to matching the value in one field. However, you can apply a second match to the entries that are displayed, to create a smaller selection, and build a compound match condition.

There are three ways of matching: cursor-selecting a matching value, over-typing existing values, or using the MATCH command. Each of these are explained separately in the following.

Each of these are integrated with the Column Attributes display and updates the MATCH field of this display as if a value had been typed there initially.

### Cursor-selecting a matching value

The first way of matching is to move the cursor under the value you want matched, then press PF4 (MatchCSR). Fault Analyzer refills the fault history window with faults that share this value for this field.

For example, if on the sample screen shown at Figure 6 on page 35 you moved the cursor under the Abend value on the last visible entry and pressed PF4, the new display shows only those faults that had an abend of S0CB. The resulting list of entries might look like this:

The screenshot shows the IBM Fault Analyzer interface with the following details:

- Menu Bar:** File, Options, View, Services, Help
- Title Bar:** IBM Fault Analyzer - Fault Entry List
- Status Bar:** 1 3 of 319 rows matched, Scroll ==> CSR
- Command Line:** Command ==> \_\_\_\_\_
- Fault History File or View:** 'IBMUSER.DEMO.HIST'
- Message Area:** {The following line commands are available: ? (Query), V or S (View saved report), I (Interactive reanalysis), B (Batch reanalysis), D (Delete), H (Duplicate history), C (Copy fault entry), M (Move fault entry), X (XMIT fault entry)..}
- Data Table:**

Fault ID	Job/Tran	User ID	Sys/Job	Abend	Date	Time
F00294	DB2NL2	IBMUSER	MVS2	S0CB	2001/02/20	14:42:29
F00292	DB2LE2	IBMUSER	MVS2	S0CB	2001/02/20	14:38:25
F00049	DACBB001	JCULLEN	MVS2	S0CB	2001/02/01	08:56:27
- Message:** \*\*\* Bottom of data.
- Function Keys:**
  - F1=Help F3=Exit F4=MatchCSR F5=RptFind F6=Actions F7=Up
  - F8=Down F10=Left F11=Right F12=MatchALL

Figure 15. The Fault Entry List after one match

Notes:

- 1 A message is issued that shows how many rows, of the ones previously displayed, that matched the selected value. This message only remains until a function key or the Enter key is pressed.

## The Fault Entry List display

- 2** Column headings on which a MATCH is currently active are highlighted. These remain highlighted until the MATCH is reset.

If you now move the cursor under the User ID value on the second entry and press PF4, then the new display looks like this:

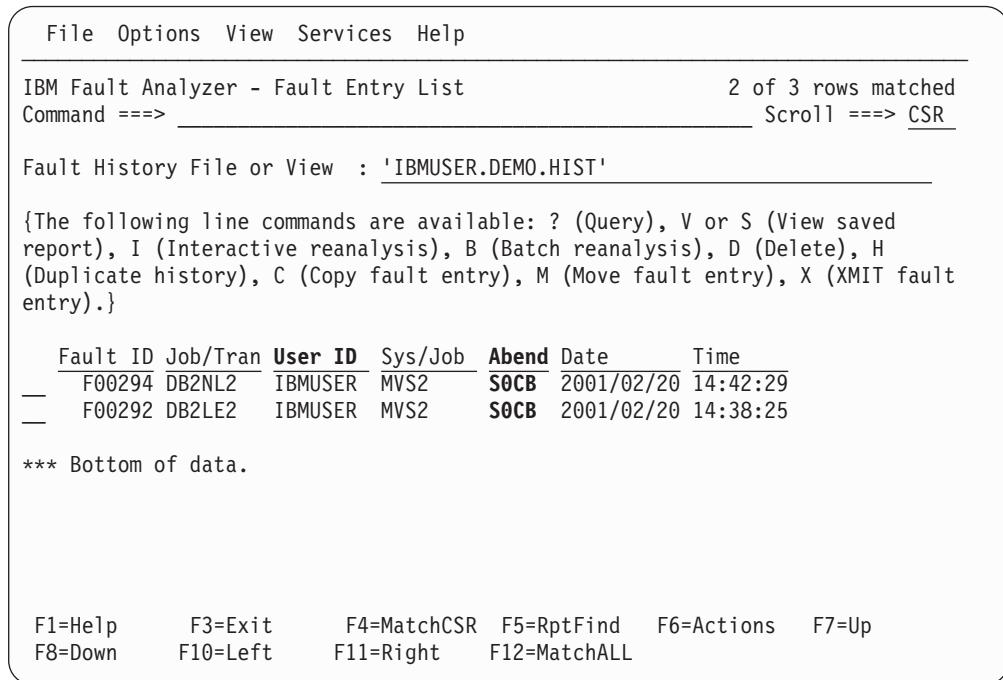


Figure 16. The Fault Entry List after two matches

Matching is restrictive. If you apply a second match, the selection of faults is restricted to those faults that have already satisfied the first match. For example, if you match by a userid, and then match by a dump status, the resultant display shows only those entries for one owner with a particular status. If instead, you just matched by status, the display shows all the entries with this status for all user IDs.

### Over-typing existing values

The second way of matching is by over-typing existing values. For example, if the Abend column contains the value S0C4 for a fault entry, then by over-typing the 4 with a 1, making the value S0C1, and pressing the Enter key, a MATCH is performed to show only those fault entries that have an abend value of S0C1.

Wildcard characters can be used to specify generic MATCH values. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate only a single character.

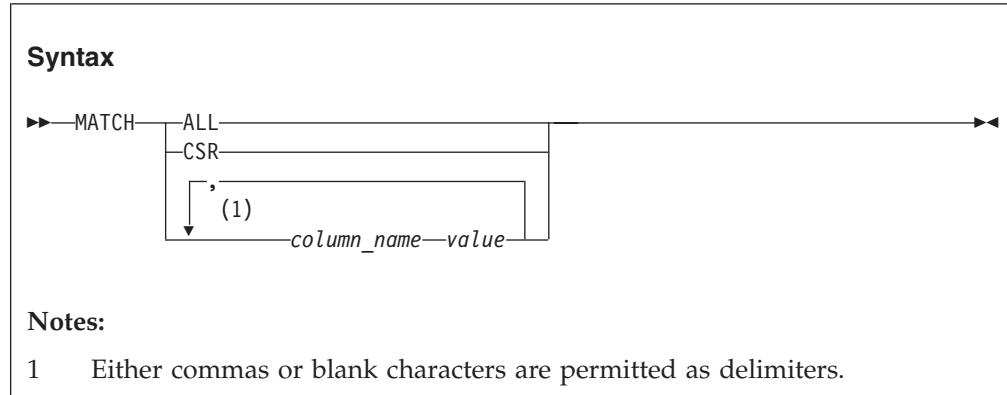
Any number of values can be over-typed before pressing the Enter key. However, if values for the same column are over-typed on multiple rows, then only the last over-typed value for that column is used.

Over-typing an existing value is particularly useful when matching on values that are similar to ones already displayed. By simply changing the displayed value to the desired target, and pressing the Enter key, a MATCH is performed with a minimum of typing required.

The value TODAY can be used in date columns to MATCH on the current date.

### Using the MATCH command

The third way of matching is by entering the MATCH command. This command has this syntax:



In the above syntax diagram, *column\_name* can be any one of the column names shown in "Available columns" on page 45, for example, Abend or IMS\_Pgm. The column name and the specified value are not case sensitive.

MATCH ALL (which is the same as PF12) removes all match conditions including the limit, so that the display shows the same entries as it had when you first displayed the history file. This option is not the same as doing a REFRESH, which looks at the history file, and so can display new entries that have been written to the history file since you first started Fault Analyzer. When you refresh, you also remove all match conditions.

MATCH CSR is the cursor match. To make this work you have to move the cursor under a value and press Enter, which is essentially the same as placing the cursor and pressing PF4 (see "Cursor-selecting a matching value" on page 49).

The other keywords correspond to a field, and you follow the field name with a value. When matching, values are not case-sensitive.

If a value contains blanks, commas or double quotes (""), then enclose it within double quotes. Any double quotes within the quoted string need to be doubled up. For example, if a displayed user title is:

A "B",C

then specify the match command as:

MATCH USER\_TITLE "A ""B""",C"

An \* can be used as a wildcard. When you append it to a value, Fault Analyzer matches all values starting with the value you entered before the \*. Since all values are strings, you could, for example, enter

MATCH DATE 2000/07\*

which would display all entries for the month of July, for 2000.

Another supported wildcard character is a percent sign (%), which can be used to indicate a single required character.

## The Fault Entry List display

The value TODAY can be used in date columns to MATCH on the current date.

You do not have to be able to see a value to enter it as part of a MATCH command. That is, you can MATCH on column values that are in the visible area of the screen, as well as column values that are outside of the current scroll window. However, you can only MATCH on columns that are currently selected for display.

If you apply a match value, and no entries satisfy this value, then Fault Analyzer displays the message "No matches", and shows a display with no entries.

## Applying an action to a particular fault

You can apply an action to a particular fault by entering a line command against the entry. Here are the available actions:

### B - batch reanalysis

Submit a batch job to reanalyze the selected fault entry. The analysis report is written to SYSPRINT.

See Chapter 4, "Performing batch reanalysis," on page 95 for details.

### C - copy

Copy the fault entry to a different history file.

See "Copying history file entries" on page 90 for details.

### D - delete

Delete the fault entry from the history file. After you delete an entry, it is immediately removed from the Fault Entry List display, and is not displayed by any subsequent refresh.

See "Deleting history file entries" on page 80 for details.

### H - duplicate history

When available, shows details about existing faults that were deemed duplicates of the selected fault entry.

If duplicate details are available for a fault entry, then the Dups column value the entry becomes a point-and-shoot field. In this case, entering the H line command against a fault entry is equivalent to placing the cursor in the Dups column value for the entry, and pressing Enter.

See "Viewing the fault entry duplicate history" on page 87 for details.

### I - interactive reanalysis

Run interactive reanalysis against the selected fault. After a little while, the interactive report is displayed. The interactive report does not replace the real-time analysis report.

See Chapter 5, "Performing interactive reanalysis," on page 103 for details.

### M - move

Move the fault entry to a different history file.

See "Moving history file entries" on page 91 for details.

### V (or S) - view report

View the saved fault analysis report:

See "Viewing a saved report" on page 74 for details.

### X - XMIT

XMIT the fault entry to a specified user ID and node.

See “Transmitting history file entries” on page 91 for details.

### ? - view fault entry information

View the fault entry information. In particular, this information shows the associated MVS™ dump data set name, if there is one.

See “Viewing fault entry information” on page 82 for details.

If you enter a line command against an entry, and Fault Analyzer is unable to complete the command, then the line command is not cleared from the line. Here are some examples of this situation:

- You attempt to run a batch dump reanalysis against a fault that has no associated dump data set.
- The dump data set is unavailable.

You can type line commands against many entries before you press Enter. In this case, Fault Analyzer attempts to honor each command, starting with the entry at the top. When Fault Analyzer is unable to honor a command, here is what happens:

- It stops processing.
- It clears the line commands from each entry it was able to process.
- It leaves the line commands for each entry it failed to process or the entry at which it was unable to honor the command.

## History file properties

To display attributes and statistics for the currently selected fault history file, first select the File menu Fault History File Properties option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens the Fault History File Properties display as the example shown in Figure 17.

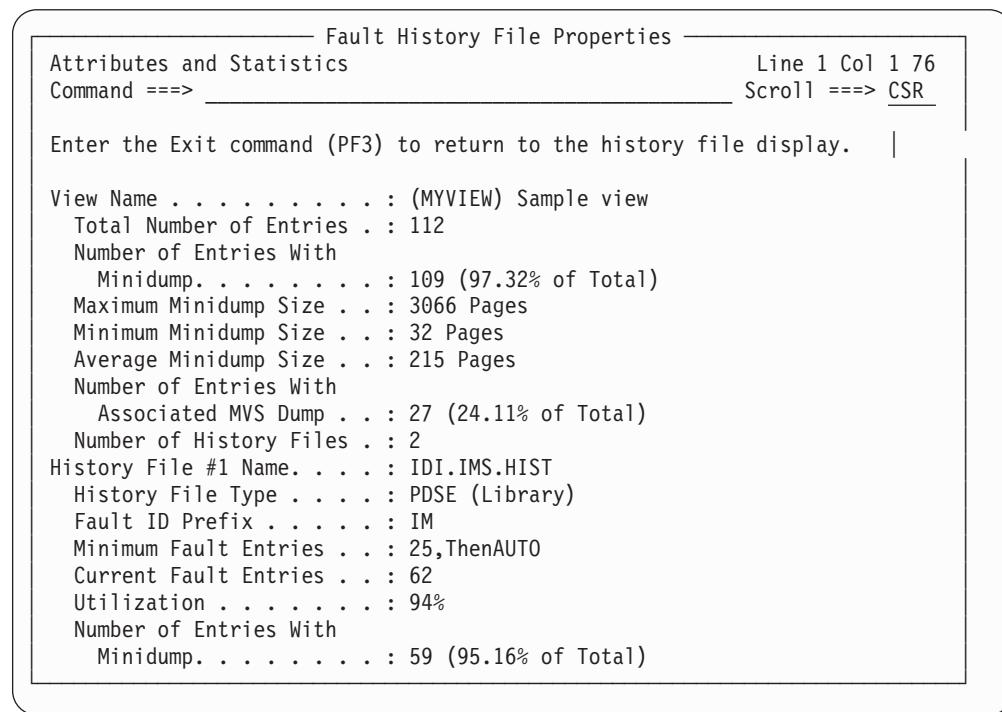


Figure 17. Sample Fault History File Properties display

The following information is available from this display:

## The Fault Entry List display

### Average Minidump Size

The average size of all minidumps saved in this history file or view is indicated as the number of minidump pages.

### Current Fault Entries

The current number of fault entries in the history file.

### Fault ID Prefix

The fault ID prefix assigned to all fault entries in a history file. This prefix can be changed with the IDIUTIL batch utility using the SETFAULTPREFIX control statement (see Chapter 27, “Managing history files (IDIUTIL utility),” on page 337). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see “Change fault history file settings” on page 57).

### History File Access

Read or Update.

### History File Name

The name of a history file explicitly displayed, or a history file that is contained in a view.

### History File Type

This name is indicated as one of the following:

- PDSE (Library)
- PDS (Partitioned Data Set)

### Minimum Fault Entries

The number of fault entries that must exist in the PDSE history file before it is maintained automatically, regardless of how many data set extents have been allocated to achieve this. Once the history file is maintained automatically, then the number of fault entries is limited only by the currently available data set space. No more data set extents are generally allocated and no out-of-space conditions are expected.

The minimum number of fault entries can be changed with the IDIUTIL batch utility SetMinFaultEntries control statement (see Chapter 27, “Managing history files (IDIUTIL utility),” on page 337). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see “Change fault history file settings” on page 57).

### Maximum Fault Entries

The maximum number of fault entries to be maintained in this history file, before automatic deletion of the oldest entries.

Additional data set extents are allocated as needed to achieve this number of fault entries. An out-of-space condition occurs if there is no space available in the data set (that is, the maximum number of data set extents has been reached or the volume is full) prior to the history file containing *nnn* fault entries.

If “n/a”, then no maximum fault entries has been assigned to the history file.

For PDS history files, the maximum number of fault entries can be changed with the IDIUTIL batch utility SetMaxFaultEntries control statement, and for PDSE history files, the IDIUTIL batch utility SetMinFaultEntries control statement (see Chapter 27, “Managing history files (IDIUTIL utility),” on page 337). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see “Change fault history file settings” on page 57).

### Maximum Minidump Size

The largest minidump saved in this history file or view is indicated as the number of minidump pages.

### Minimum Minidump Size

The smallest minidump saved in this history file or view is indicated as the number of minidump pages.

### Number of Entries With Associated MVS Dump

The number of entries in the history file or view with an associated MVS dump data set. The percentage of the total entries is also shown.

### Number of Entries With Minidump

The number of entries in the history file or view that include a saved minidump. The percentage of the total entries is also shown.

### Number of History Files

If a view is displayed, the number of history files that are contained in the view. Information about individual history files follows.

### Total Number of Entries

The total number of entries in the view.

### Utilization

Pages used as percentage of pages allocated. Only available for PDSE history files.

### View Name

If a view is displayed, the name of the view.

## New history file allocation

To allocate a new history file, first select the File menu New Fault History File Allocation option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens the New History File Allocation display as the example shown in Figure 18 on page 56.

## The Fault Entry List display

New History File Allocation

Command ==> \_\_\_\_\_ Line 1 Col 1 76  
Scroll ==> CSR

Press Enter to allocate new history file, or press PF3/PF12 to cancel.

History File Name . . . . :  
Primary Space . . . . . : 10 Cylinders  
Secondary Space . . . . . : 25 Cylinders  
Fault Entry Prefix. . . . . : F (1-3 alphabetic characters)  
Data Set Type . . . . . : LIBRARY (LIBRARY or PDS)  
Minimum Fault Entries . . . : 100 (25-9999999)

\*\*\* Bottom of data.

F1=Help F3=Exit F5=RptFind F7=Up F8=Down F12=Cancel

Figure 18. Sample New History File Allocation display

The following fields are provided on this display:

### History File Name

The history file name to be allocated. If this name is not enclosed in single quotes, then the current TSO prefix is used as the high-level qualifier.

**Note:** If this display is shown as a result of having specified a history file that does not exist for a Fault Entry List Move or Copy line command, then the History File Name field is not available for input, but shows the Move/Copy target history file name.

### Primary Space

The number of cylinders to allocate as the primary space. The default is 10.

### Secondary Space

The number of cylinders to allocate as the secondary space. The default is 25.

### Fault Entry Prefix

A one to three character prefix assigned to all fault IDs in this history file. The default is F.

### Data Set Type

The type of history file data set to allocate as one of the following:

- LIBRARY (PDSE)
- PDS (Partitioned Data Set)

The default is LIBRARY.

### Minimum Fault Entries

The minimum number of fault entries that must exist in the history file before automatic space management occurs. This number must be between 25 and 9999999. The default is 100.

**Note:** This field is only shown if the Data Set Type field specifies LIBRARY.

### Maximum Fault Entries

The maximum number of fault entries to maintain in the history file as a number between 25 and 9999999. The default is 100.

**Note:** This field is only shown if the Data Set Type field specifies PDS.

### Change fault history file settings

To change a history file fault ID prefix or minimum/maximum number of fault entries, first ensure that the history file is selected on the Fault Entry List display, or is included in the currently selected View. Then, select the File menu Change Fault History File Settings option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61).

If a View is currently selected, then an option is provided to select a history file from the list of history files that are contained in the View.

Next, the Change Fault History File Settings display, as the example shown in Figure 19, is presented.

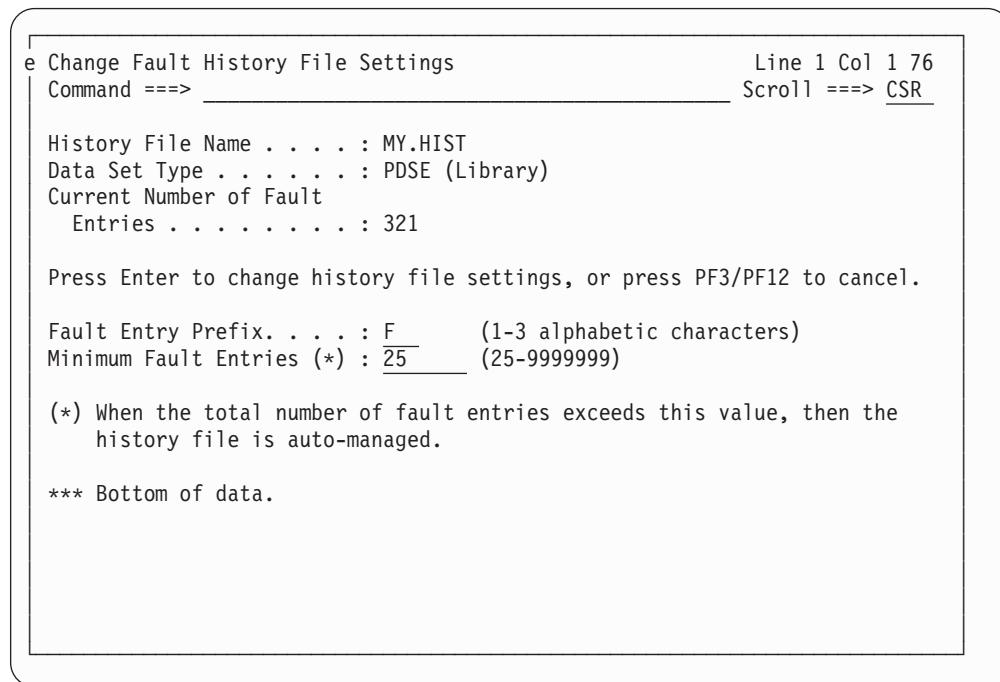


Figure 19. Sample Change Fault History File Settings display

The following fields are provided on this display:

#### History File Name

The history file name whose settings are to be changed.

#### Data Set Type

The selected history file type as one of the following:

- PDSE (Library)
- PDS

#### Current Number of Fault Entries

The number of fault entries currently in the selected history file.

#### Fault Entry Prefix

A one to three alphabetic character prefix which, together with the

## The Fault Entry List display

appended fault entry number, forms the fault ID for this history file. All new fault entries created in the history file are given this prefix, and by using different prefixes for different history files, this can make the fault IDs easier to recognize.

The initial fault entry prefix value shown always reflects the current setting for the history file.

### Minimum Fault Entries

The minimum number of fault entries that must exist in the PDSE history file before automatic space management occurs. This number must be between 25 and 9999999.

**Note:** This field is only displayed if the Data Set Type field specifies PDSE (Library).

### Maximum Fault Entries

The maximum number of fault entries to maintain in the PDS history file as a number between 25 and 9999999.

**Note:** This field is only displayed if the Data Set Type field specifies PDS.

The initial minimum or maximum number of fault entries value shown typically reflects the current setting for the history file. However, in the following cases, the value is instead a recommended value:

- The history file is a PDS and no maximum number of fault entries has been set. In this case, the initial maximum number of fault entries value is set to 100.
- The history file is a PDSE and no minimum number of fault entries has been set. In this case, the initial minimum number of fault entries value is set to 100.
- The history file is a PDSE and the minimum number of fault entries has either not been set, or is less than 25. In this case, the initial minimum number of fault entries value is set to 25.

A PDSE history file is always enabled for auto-management if changing the minimum number of fault entries value, regardless of whether it was auto-managed before or not.

After changing any settings, press Enter to save.

To exit without saving, press PF3 or PF12.

The functionality provided by this display is equivalent to the use of the IDIUTIL batch utility SetFaultPrefix, SetMaxFaultEntries and SetMinFaultEntries control statements.

**Note:** The action-bar option "File->Change Fault History File Settings..." is not selectable if the user's administrator authorization to the history file currently being displayed, or to all history files in the current View, has been restricted. For details about restricting authorization, see "Restricting change of history file settings" on page 233.

## Resetting history file access information

To reset all information about previously accessed history files or views, and previously accessed history file entries, select the File menu Clear Last Accessed Information option (for information about selecting this option in general, refer to "Action-bar pull-down menus" on page 61).

Immediately after selecting this option, no entries are available when selecting the File menu Last Accessed Fault History File Entries option. However, the File menu Last Accessed Fault History Files or Views option shows a single entry for the currently active history file or view.

### Refreshing fault entry information

While displaying a history file or view, it is possible that new entries are being added, for example due to real-time analysis of abending jobs. To reread the history file or view to include any such entries, you can either issue the REFRESH command or select the View menu Refresh option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61).

The refresh performed in this manner causes the display to be reformatted and positioned at the top-most line and left-most column. Any MATCH command filtering that was active at the time of the refresh is reset.

#### Implicit refresh

Implicit refresh is performed when the Enter key is pressed from the Fault Entry List display, without entering a primary command, a line command, or overwriting any column data, the list of fault entries is reread from the history file or view.

#### Updates pending

To avoid a potentially lengthy response time when initially displaying or refreshing the Fault Entry List display, only history file information that is immediately available is displayed.

An example of a history file that is not immediately available is one that is in use on a different MVS image to the one on which it is being displayed. In this case, the information displayed is limited to the fault entries currently in the \$\$INDEX member of the history file data set, but does not include any new or modified faults that might have been added to the still cached \$\$INDEX member which is managed by the IDIS subsystem on the other MVS image. As soon as the other IDIS subsystem relinquishes control of the history file, the updated information is available for display.

If one or more history files are not immediately available, then a message is displayed near the top of the display as shown in the following example (1):

## The Fault Entry List display

IBM Fault Analyzer - Fault Entry List								Refresh	complete
Command ==> _____								Scroll	==> CSR
<b>Fault History File or View : (ALLTEST) All testing</b>									
<b>6 history files might have updates pending. [1]</b>									
Fault_ID	MD_Pages	Dups	Dup_Date	Dup_Time	Abend	I_Abend	Module	System	
IMS14434	96		2006/08/03	13:51:52	S0CB	U4039	IDCB0060	FAE1	
IMS14433	99		2006/08/03	13:51:40	SNAP	SNAP	IDCB0040	FAE1	
IMS14432	98		2006/08/03	13:51:26	S0CB	U4039	IDCB0020	FAE1	
IMS14431	51		2006/08/03	13:51:16	U0428	U0428	DFSPCC20	FAE1	
IMS14430	98		2006/08/03	13:51:01	S0CB	U4039	IDCB0020	FAE1	
IMS14429	99		2006/08/03	13:50:46	S0CB	U4039	IDCB0010	FAE1	
IMS14428	106		2006/08/03	13:50:33	S0CB	U4039	IDCB0090	FAE1	
IMS14427	735		2006/08/03	13:50:11	S0CB	U4039	IDCB0080	FAE1	
IMS14426	105		2006/08/03	13:49:54	S0CB	U4039	IDCB0070	FAE1	
IMS14424	99		2006/08/03	13:49:35	S0CB	U4039	IDCB0060	FAE1	
IMS14423	51		2006/08/03	13:49:26	U0456	U0456	DFSPCC20	FAE1	
IMS14422	100		2006/08/03	13:49:15	SNAP	SNAP	IDCB0040	FAE1	
IMS14421	98		2006/08/03	13:49:02	S0CB	U4039	IDCB0020	FAE1	
IMS14420	96		2006/08/03	13:48:48	S0CB	U4039	IDCB0020	FAE1	
IMS14419	51		2006/08/03	13:48:39	U0428	U0428	DFSPCC20	FAE1	
IMS14418	97		2006/08/03	13:48:26	S0CB	U4039	IDCB0010	FAE1	

Figure 20. Sample Fault Entry List display with updates pending

By placing the cursor on the number of history files in the updates pending message ([1]), and pressing Enter, the History File Updates Pending display is shown as in the following example:

History File Updates Pending								Line 1 Col 1 76
Command ==> _____								Scroll ==> CSR
The most current information from the following history files might not be included--press PF3, PF12, or Enter to continue, and refresh later by pressing Enter again:								
CTEST.DANLEPLI.DCAT CTEST.DAPLRMVS.DCAT CTEST.DAQJSMVS.DCAT CTEST.DAQSCMVS.DCAT CTEST.DAQSOMVS.DCAT								
IMS14427 735 2006/08/03 13:50:11 S0CB U4039 IDCB0080 FAE1 IMS14426 105 2006/08/03 13:49:54 S0CB U4039 IDCB0070 FAE1 IMS14424 99 2006/08/03 13:49:35 S0CB U4039 IDCB0060 FAE1 IMS14423 51 2006/08/03 13:49:26 U0456 U0456 DFSPCC20 FAE1 IMS14422 100 2006/08/03 13:49:15 SNAP SNAP IDCB0040 FAE1 IMS14421 98 2006/08/03 13:49:02 S0CB U4039 IDCB0020 FAE1 IMS14420 96 2006/08/03 13:48:48 S0CB U4039 IDCB0020 FAE1 IMS14419 51 2006/08/03 13:48:39 U0428 U0428 DFSPCC20 FAE1 IMS14418 97 2006/08/03 13:48:26 S0CB U4039 IDCB0010 FAE1								

Figure 21. Sample History File Updates Pending display

This display lists the names of all history files for which the most current information might not have been available. To return from this display, press PF3, PF12, or Enter.

If the updates pending message is received, and the most current information is required, then this information is generally shown when the Enter key is pressed again to perform an implicit refresh.

### Fault entry expiration control

To prevent premature deletion of fault entries, these can be locked either indefinitely, or for a specified number of days following the initial creation of a fault entry. Either way, this deletion is controlled via the Lock Flag, which can be viewed or modified using the Fault Entry Information display (for details, see “Viewing fault entry information” on page 82).

The Lock Flag can also be set by a user exit via the ENV.LOCK\_FLAG field (for details, see “ENV - Common exit environment information” on page 513).

---

### Action-bar pull-down menus

Most of the displays used by the Fault Analyzer ISPF interface include an action-bar located at the top of the panel. The ACTIONS ISPF command (by default mapped to PF6 on some displays) can be used to place the cursor at the left-most action available. Depending on ISPF settings, you might then be able to move the cursor to other actions by pressing the Tab key. Alternatively, you can simply use the Up/Down/Left/Right Arrow keys to place the cursor on the action of your choice. Using a PF key to issue the ACTIONS command is advantageous as the cursor is automatically repositioned in the display at the location where it was before the action was selected.

Once the cursor is placed on an action-bar item, press the Enter key to show the associated pull-down menu.

Options in pull-down menus can be selected by either entering the associated option number at the initial cursor position, or by placing the cursor (using the Up/Down/Left/Right Arrow keys) anywhere on the line of the option and pressing the Enter key. Any options not available for selection are indicated by an asterisk (\*) instead of a numerical option number.

In the following list of available pull-down menu options, the format used is:  
menu\_name->menu\_option->menu\_option...

where

**menu\_name**

The name of the action-bar pull-down menu at the top of the display.

**menu\_option**

The name of the available option on the first and any subsequent menus.

The available pull-down menu options are show below in alphabetic order:

**File->Analyze MVS Dump Data Set**

Used to initiate interactive analysis of a SYSMDUMP or SVC dump data set—primarily intended for CICS system dump analysis (see Chapter 6, “Performing CICS system abend dump analysis,” on page 177), or Java dump analysis (see Chapter 8, “Performing Java analysis,” on page 189).

## Action-bar pull-down menus

### **File->Change Fault History File Settings**

Used to change the fault ID prefix or maximum number of fault entries history file settings. See “Change fault history file settings” on page 57 for more information.

### **File->Clear Last Accessed Information**

Used to reset information about previously accessed history files or views, and previously accessed history file entries. See “Resetting history file access information” on page 58 for more information.

### **File->Exit**

Selecting this option is the equivalent to issuing the ISPF EXIT command. Generally, this command ends the current display and returns to the display from which it was invoked.

### **File->Exit Fault Analyzer**

Used to exit from the Fault Entry List display. Selecting this option is equivalent to issuing the EXIT command (or pressing the PF3 key).

### **File->Exit Interactive Reanalysis**

Used to return from anywhere in the interactive reanalysis report to the Fault Entry List display. Selecting this option is **not** equivalent to issuing the EXIT command (or pressing the PF3 key), as the EXIT command only returns to the previous display.

### **File->Fault Entry Information**

Used to open the Fault Entry Information display as shown in Figure 29 on page 83. Selecting this option is identical to issuing the INFO command (see “INFO” on page 69) or entering the “?” line command against a fault entry from the Fault Entry List display (see “Applying an action to a particular fault” on page 52).

### **File->Fault History File Properties**

Used to display attribute and statistical information pertaining to the currently selected history file or view. See “History file properties” on page 53 for more information.

### **File->Format CICS Auxiliary Trace Data Set**

Used to format a CICS auxiliary trace data set. See Chapter 7, “Formatting a CICS auxiliary trace data set,” on page 187 for more information.

### **File->Last Accessed Fault History File Entries**

Used to display a list of up to 10 previously accessed history file entries. See “Changing the history file or view displayed” on page 37 for more information.

### **File->Last Accessed Fault History Files or Views**

Used to display a list of up to 10 previously accessed history files or views. See “Changing the history file or view displayed” on page 37 for more information.

### **File->List Views**

Used to display a list of all available views. See “Changing the history file or view displayed” on page 37 for more information.

### **File->New Fault History File Allocation**

Permits the allocation of a new history file. See “New history file allocation” on page 55 for additional information.

### **Help->About Fault Analyzer**

Used to display copyright and general usage information for Fault

Analyzer. See “Displaying Fault Analyzer copyright and general usage information” on page 80 for more information.

**Options->Batch Reanalysis Options**

Used to set options for batch reanalysis. See “Batch reanalysis options” on page 95 for more information.

**Options->Fault Analyzer Preferences**

Used to set options that affect the behavior of the Fault Entry List display.

**Options->Interactive Reanalysis Options**

Used to set options for interactive reanalysis. See “Interactive reanalysis options” on page 103 for more information.

**Services->Copy Current Display to Data Set**

Used to copy the entire contents of the current display to a data set. See “Copying interactive displays to a file” on page 79 for more information.

**Services->COBOL Explorer**

Used to start the COBOL Explorer dialog with prompt for program selection. See “COBOL Explorer” on page 172 for more information.

**Services->IDILANGP Side File Formatting Utility**

Used to invoke the IDILANGP side file formatting utility to display a pseudo compiler listing. See *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* for more information.

**Services->List User Notes**

Used to show all user notes that exist for the current fault entry. Selecting this option is the equivalent to issuing the NOTELIST command. See “Creating and managing user notes” on page 150 for more information.

**Services->Message ID Lookup**

Used to show the explanation for a user-selected message, abend code, or other information. See “Displaying user-selected message or abend code explanations” on page 77 for more information.

**View->Add Blank Lines**

Used to format displays by using blank lines when appropriate to separate the information that is shown. This value is the default. See “Adding or removing blank lines” on page 76 for more information.

**View->Add Help Text**

Used to add explanatory text to displays which might assist the inexperienced user. This value is the default. See “Adding or removing help text” on page 76 for more information.

**View->Add Pseudo Assembler Instructions**

Used to add pseudo-assembler instructions to the Compiler Listing display. See “Displaying source code” on page 145 for more information.

**View->Column Configuration**

Used to change the columns of information that are shown for faults listed on the Fault Entry List display. See “Fault entry list column configuration” on page 43 for more information.

**View->Preferred formatting Width**

Used to set the preferred display formatting width. See “Setting preferred formatting width” on page 76 for more information.

## Action-bar pull-down menus

### **View->Refresh**

Used to reread all entries in the selected history file or view. See “Refreshing fault entry information” on page 59 for more information.

### **View->Remove Blank Lines**

Used to eliminate blank lines as much as possible in displays to make the maximum amount of information visible on screens capable of only showing a small number of lines. See “Adding or removing blank lines” on page 76 for more information.

### **View->Remove Help Text**

Used to remove explanatory text from displays. See “Adding or removing help text” on page 76 for more information.

### **View->Remove Pseudo Assembler Instructions**

Used to remove pseudo-assembler instructions from the Compiler Listing display. This value is the default. See “Displaying source code” on page 145 for more information.

---

## Commands

From the Fault Analyzer ISPF interface, the following primary commands are available:

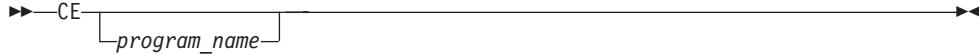
**Note:** Not all commands can be issued from all displays. Refer to the description of individual commands for details.

### **CE**

Invokes COBOL Explorer. COBOL Explorer asks for the variables on the event source line that are to be used for the branch analysis. The branch analysis is a procedure traceback that shows the use of the selected variables.

Enter this command during interactive reanalysis.

#### Syntax



```
➡➡CE [program_name]
```

#### *program\_name*

The name of the associated COBOL program event. If this name is omitted, a pop-up list of COBOL program events is shown.

For more information, see “COBOL Explorer” on page 172.

### **COLS**

Brings up the Fault Entry List Column Configuration display that allows tailoring of the information that is shown on the Fault Entry List display.

This command is only available from the Fault Entry List display.

**Syntax**

```
►►COLS—————►►
```

For more information, see “Fault entry list column configuration” on page 43.

**COPY**

Copies the current display to the specified data set name.

**Syntax**

```
►►COPY [data_set_name]—————►►
```

where:

**data\_set\_name**

The data set to which the display is to be written.

The standard TSO naming convention for data sets is assumed, that is, if the data set name is not enclosed in single quotes, the current TSO prefix is used as the high-level qualifier. If the data set is partitioned, a member name must also be specified in parenthesis after the data set name.

If no data set name is specified, a pop-up panel is displayed from which the data set name can be entered.

If a sequential data set name is specified, and the data set does not exist, then it is created using the attributes: RECFM=VB LRECL=1028 BLKSIZE=6144. The attributes of any existing data sets are not modified.

The copied data is truncated if the logical record length of the data set is insufficient.

For more information about the COPY command and an example of its use, see “Copying interactive displays to a file” on page 79.

**DISASM**

This command, which is only available from within the interactive report, can be used to perform disassembly of object code at a given address in storage.

**Syntax**

```
►►DISASM—————►►
```

For information about how to use this command, see “Disassembling object code” on page 160.

## DSECT

### DSECT

This command, which is only available from within the interactive report, can be used to provide formatting of storage areas based on user-supplied assembler macro or DSECT copybooks.

#### Syntax

```
►►DSECT►►
```

For information about how to use this command, see “Mapping storage areas using DSECT information” on page 154.

## DUPS

This command, which is only available from within the interactive report, can be used to show details about duplicate faults that have occurred against the current fault entry.

#### Syntax

```
►►DUPS►►
```

For information about the display of duplicate fault details, see “Viewing the fault entry duplicate history” on page 87.

## EXEC

This command, which is only available from within the interactive report, can be used to invoke a Formatting user exit.

#### Syntax

```
►►EXEC►►
```

```
[exit_name [parameters]]
```

where:

#### exec\_name

The name of the Formatting user exit to be executed.

If no exit name is specified, then a display is presented from which an exit can be selected.

#### parameters

Are optional parameters to be passed to the Formatting user exit.

If multiple blank-delimited parameters are specified, then these are passed to the user exit separately.

For example, if a user exit MYEXEC accepts two parameters, "A" and "B", then the exit is invoked from the ISPF command line as:

```
EXEC MYEXEC A B
```

and the parameters can be received in the exit using:

```
PARM1 = ARG(1)
PARM2 = ARG(2)
```

Note that

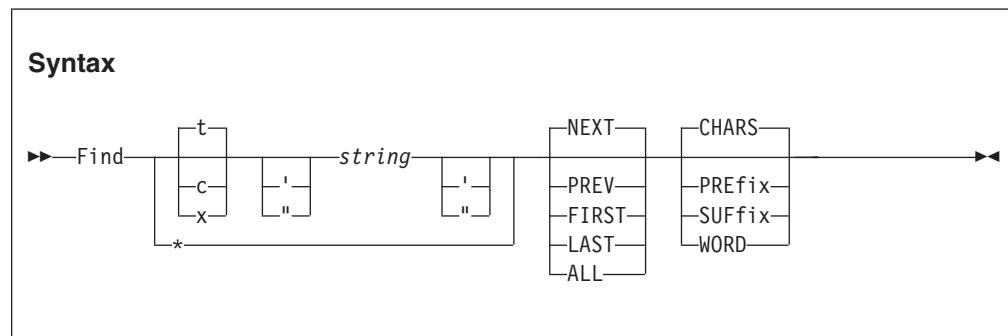
```
PARSE ARG PARM1 PARM2
```

will not work as expected.

For information on creating your own Formatting user exits, and making these available to your environment, see “User-specific report formatting” on page 164.

## FIND

Used to locate a text string in the current display.



where:

- t** Indicates that a text string is being searched for. Text strings are non-case-sensitive in all displays, except for the Dump Storage display where all strings are case-sensitive.
- c** Indicates that a character string is being searched for. Character strings are case-sensitive.
- x** Indicates that a hexadecimal value is being searched for. An even number of digits must be specified.
- string** The character string to be searched for. If the string includes blanks, quotes, or tokens that could be mistaken for a FIND command keyword (for example, if searching for the string NEXT), the string must be enclosed in either single quotes or double quotes. The ending quote must be of the same type (single or double) as the starting quote—all other quotes are considered part of the search string.
- \*** Indicates that the same string as previously searched for is to be used.
- NEXT** Indicates that the search should start at the first position after the current cursor location and proceed ahead to find the next occurrence of the string. NEXT is the default.
- PREV**
- FIRST**
- LAST**
- ALL**
- CHARS**
- PREFIX**
- SUFFIX**
- WORD**

## FIND

<b>PREV</b>	Indicates that the search should start at the first position before the current cursor location and proceed backwards to find the previous occurrence of the string.
<b>FIRST</b>	Indicates that the search should start at the top of the display and proceed ahead to find the first occurrence of the string.
<b>LAST</b>	Indicates that the search should start at the bottom of the display and proceed backwards to find the last occurrence of the string.
<b>ALL</b>	Indicates that the search should start at the top of the display and proceed ahead to find all occurrences of the string. A message in the upper-right corner of the display shows the number of occurrences found.
<b>CHARS</b>	Indicates that any occurrence of the sequence of characters searched for is considered a match. This value is the default.
<b>PREFix</b>	Indicates that one or more blank or attribute characters must precede the sequence of characters searched for in order to be considered a match. Either PRE or PREFIX can be specified.
<b>SUFfix</b>	Indicates that one or more blank or attribute characters must follow the sequence of characters searched for in order to be considered a match. Either SUF or SUFFIX can be specified.
<b>WORD</b>	Indicates that one or more blank or attribute characters must surround the sequence of characters searched for in order to be considered a match.

### FIND command differences between display types

The FIND command that is issued from the Dump Storage display behaves differently to the FIND command that is issued from all other displays. (The Dump Storage display is invoked by using the SHOW command, or by placing the cursor on an address point-and-shoot field and pressing Enter.) The differences are explained in the following comparison of the displays:

*Table 4. FIND command differences between display types*

FIND command in Dump Storage display	FIND command in all other displays
All character strings are case sensitive, regardless of whether T'text' or C'text' or neither is used.	Character string case sensitivity is determined by the use of T'text' (default) or C'text'.
Only the minidump, and any associated MVS dump, is searched for the target string—storage descriptions, headings and similar formatted character-based text is not included in the search. However, the entire minidump and MVS dump is included in the search—not just the currently displayed address range.	Only the formatted character-based text, including any hex-dump formatting, is searched for the target string—no searching of the minidump or MVS dump is performed.
When searching for hexadecimal values in the minidump, use the X'text' format.	When searching for values in hex-formatted storage, use character string format.
The FIND command target can be split over multiple lines in the formatted display, but can still be found.	The FIND command target cannot be found if split over multiple lines in the formatted display.

## INFO

This command, which is only available from within the interactive report, can be used to view information about the current fault entry.

Issuing the INFO command is identical to selecting "Fault Entry Information" from the interactive reanalysis report action-bar "File" pull-down menu (see "Action-bar pull-down menus" on page 61), or entering the "?" line command against a fault entry from the Fault Entry List display (see "Applying an action to a particular fault" on page 52). In either case, the Fault Entry Information display, as shown in Figure 29 on page 83, is presented.

### Syntax

```
►►—INFO—►►
```

## LOOKUP

Displays the explanation of a specified message ID, abend code, or other information.

### Syntax

```
►►—LOOKUP—  
          |  
          arg
```

where:

**arg**      The message ID, abend code, or other item of information to be retrieved.

If *arg* is not specified, then a pop-up panel is displayed from which the required information can be selected or searched for.

The use of an asterisk as a wildcard character representing any number of characters is permitted. An implicit asterisk is assumed at both the beginning and at the end of *arg*, if no other asterisks are specified.

A percent sign can be used to represent only a single character.

The following are sample search arguments that illustrate the use of the wildcard characters:

- XYZ**      Implicitly searches for a match on \*XYZ\*, which means that zero or more characters might precede XYZ, and zero or more characters might follow (for example, XYZ, AXYZ, XYZB, AXXYZBBB).
- A\***      Matches everything starting with A, followed by zero or more characters (for example, A, AB, ABA000I).
- A%**      Matches everything starting with A, followed by exactly one character (for example, AA, AB, AC).

## LOOKUP

### **ABA%%%%I**

Matches everything starting with ABA, followed by 4 characters and ending with I (for example, ABA0000I).

If the search results in more than one item of information, then a list is displayed from which selections can be made.

Only information that is provided by Fault Analyzer, or specified in member IDIHUSR.M of the IDI.SIDISAM1 data set, can be entered.

Information, such as explanations for messages or abend codes, is searched for in the following places and in the order shown:

1. User-defined information in member IDIHUSR.M of the IDI.SIDISAM1 data set.  
For details, see Chapter 28, "Providing application-specific explanations and descriptions," on page 349.
2. IBM-supplied information in the IDIDOC data set.
3. IBM-supplied information in the IDIVSxxx data set, where xxx is the 3-character language ID in effect for the Language option.
4. In the case of message or abend code explanations, the Message and Abend Code Explanation user exit might choose to supply an alternate explanation.  
For details, see "Message and Abend Code Explanation user exit" on page 377.

The LOOKUP command, along with its LOOKC variant, can also be issued from outside of the Fault Analyzer ISPF interface, as long as they are issued under ISPF and the sample Fault Analyzer ISPF command table has been made available—for details, see Chapter 15, "Modifying your ISPF environment," on page 247.

For more information about the LOOKUP command, and an example of its use, see "Displaying user-selected message or abend code explanations" on page 77.

## MATCH

Refer to "Additional ways to match and select faults" on page 49 for information about the MATCH command.

## NEXT

This command is only available from within the interactive report and its behavior is dependent on the type of display from which it is invoked:

### **Event Details**

Use this command to select the next event. For example, if you are currently displaying event number 2, entering this command displays event number 3, if available.

### **Dump Storage**

The storage address displayed prior to entering a PREV command is shown again.

This command is usually assigned to the PF11 function key.

**Syntax****►►—NEXT—►►****NOTELIST**

This command, which is only available from within the interactive report, can be used to display all user notes that exist in the current fault entry.

**Syntax****►►—NOTELIST—►►**

For more information, see “Creating and managing user notes” on page 150.

**PREV**

This command is only available from within the interactive report and its behavior is dependent on which of the following types of information that is being displayed:

**Event details**

Use this command to select the previous event. For example, if you are currently displaying event number 2, entering this command displays event number 1.

**Dump Storage Display**

The previously displayed storage address (if any) is shown.

This command is usually assigned to the PF10 function key.

**Syntax****►►—PREV—►►****QUIT**

The behavior of this command depends on from where it is issued:

- From within the interactive reanalysis report, the user is returned to the Fault Entry List display.

This behavior is the equivalent to selecting "Exit Interactive Reanalysis" from the File pull-down menu.

- From the Fault Entry List display, the ISPF interface is terminated.

This behavior is the equivalent to selecting "Exit Fault Analyzer" from the File pull-down menu.

## **QUIT**

### **Syntax**

```
►►QUIT►►
```

## **REFRESH**

This command is only available from the Fault Entry List display.

Causes the current history file or view to be reread to include in the display any updates that might have been created since the initial selection of the file or view, or since the last time the REFRESH command was issued.

Any MATCH command that might be active is reset.

Issuing the REFRESH command performs the same function as when the Refresh option is selected from the View action-bar pull-down menu.

### **Syntax**

```
►►REFRESH►►
```

## **RESET**

This command is only available from the Fault Entry List Column Configuration display.

Causes the Fault Entry List column configuration to be changed to the configuration defined by the HistCols option in effect.

### **Syntax**

```
►►RESET►►
```

## **RPTFIND**

Locates the next occurrence of the search argument entered for the last FIND command. This command is by default mapped to the PF5 function key.

### **Syntax**

```
►►RPTFIND  
|  
| RF  
|
```

## RUNCHAIN

This command, which is only available from within the interactive report, can be used to display chained data areas.

### Syntax

```
►►RUNCHAIN—————►►
```

For information about how to use this command, see “Displaying chained data areas” on page 157.

## SHOW

This command, which is only available from within the interactive report, can be used to display a storage location.

### Syntax

```
►►SHOW [0] address [offset]—————►►
```

For example, to display the storage at address 007F2300, enter:

SHOW 7F2300

The address parameter is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

If the SHOW command is issued without specifying an address, then either 0, or the storage address that is associated with the display of a CICS system dump analysis data area from which the SHOW command is issued, or the last address selected for SHOW, is used as the default.

Offsets can be specified with the show command. For example:

SHOW 142A0 + 1C0 + 1B - 8

All offsets are in hex. One or more blanks must separate SHOW and the base address, whereas blanks separating any offsets are optional.

## SHOWFREE

This command can be used to check the amount of available storage in the TSO region.

## **SHOWFREE**

### **Syntax**

```
►►SHOWFREE►►
```

Sample output from the SHOWFREE command follows:

```
Largest Contiguous Virtual Storage Block Available:  
    Above the Line. . : 12.30 MB  
    Below the Line. . : 3.66 MB  
Total Virtual Storage Available:  
    Above the Line. . : 24.86 MB  
    Below the Line. . : 3.70 MB  
***
```

## **STCK**

This command, which is only available from within the interactive report, can be used to convert a binary STORE CLOCK value to a human-readable date and time format.

### **Syntax**

```
►►STCK►►
```

For information about how to use this command, see “Converting STORE CLOCK values” on page 162.

## **VIEWS**

This command, which is only available from the Fault Entry List display, can be used to show a listing of all views currently available. Issuing the VIEWS command performs the same function as when the List Views option is selected from the File action-bar pull-down menu (see “Select a view from a list of views” on page 41).

### **Syntax**

```
►►VIEWS►►
```

## **Viewing a saved report**

To view the saved report that is associated with a fault, enter **V** (or **S**) against the entry in the history file display. Figure 22 on page 75 shows a sample Saved Report display.

```

File View Services Help
Saved Report Line 1 Col 1 80
Command ==> Scroll ==> CSR
- Expand all / + Collapse all ****
* IBM Fault Analyzer for z/OS V7R1M0 (MVS 2007/01/16)
*
* (C) Copyright IBM Corp. 2000, 2007. All rights reserved.
****

JOBNAME: IDIVPPL2 SYSTEM ABEND: 0C9 FAE1 2005/06/22 09:28:

+ <H1> I B M   F A U L T   A N A L Y Z E R   S Y N O P S I S
+ <H1> I B M   F A U L T   A N A L Y Z E R   E V E N T   S U M M A R Y
+ <H1> I B M   F A U L T   A N A L Y Z E R   E V E N T   D E T A I L S
+ <H2> E V E N T 1 O F 5: C A L L (D S A A D D R E S S 0 0 0 3 4 0 1 8)
+ <H2> E V E N T 2 O F 5: C A L L (D S A A D D R E S S 0 0 0 3 4 0 E 0)
+ <H2> E V E N T 3 O F 5: C A L L (D S A A D D R E S S 0 0 0 3 4 1 D 0)
+ <H2> E V E N T 4 O F 5: C A L L (D S A A D D R E S S 0 0 0 3 4 3 9 0)
+ <H3> A s s o c i a t e d   S t o r a g e   A r e a s
F1=Help      F3=Exit      F5=RptFind     F7=Up      F8=Down      F10=Left
F11=Right

```

*Figure 22. Sample Saved Report display*

Use the scroll keys (PF7/PF8/PF10/PF11) to view the entire report.

The report viewed from the history file is normally the same as the real-time report inserted in your job output on the JES spool.

However, if no report was written to the fault entry when it was first created, then a pseudo batch reanalysis report is added whenever possible the first time an attempt is made to view it with the 'V' or 'S' line command, provided that the user has update access to the history file. Fault entries without real-time reports are those created with the DeferredReport option in effect, or recovery fault recording fault entries.

Since the report that can be viewed might be a real-time report, or might be a batch reanalysis report created and saved at a later stage, the common term "saved report" is used to refer to the report that is contained in the fault entry.

Note that the Batch Reanalysis Options, not the Interactive Reanalysis Options, are used when creating the saved report.

If an attempt is made to view the saved report for a fault entry that does not contain one, and it is not possible to create one either, then, if the fault entry contains a minidump or an associated MVS dump exists, interactive reanalysis is automatically performed instead as if the 'T' line command had been used. This process is the case for CICS system dump fault entries, or for any fault entries without a saved report in a history file to which the user does not have update access.

To enable easier navigation, individual sections of the report can be collapsed or expanded by placing the cursor on the + or - sign point-and-shoot fields preceding each report heading, and pressing the Enter key:

- If a - sign is shown, then the section is currently expanded and, if the cursor is placed on the - and the Enter key is pressed, the section is collapsed.

## **Viewing a saved report**

- If a + sign is shown, then the section is currently collapsed and, if the cursor is placed on the + and the Enter key is pressed, the section is expanded.

Only the heading line itself is visible for collapsed report sections.

At the top of the display are two +/- point-and-shoot fields that permit all report sections to be expanded or collapsed collectively. Whenever one of these is selected, the current setting is saved in the user's ISPF profile and used as the initial setting on any subsequent saved report displays.

---

## **Adding or removing blank lines**

On screens with only a small number of lines able to be displayed at once, it might be advantageous to eliminate the insertion of blank lines in Fault Analyzer displays as much as possible. That way, the information is "condensed" and the need for vertical scrolling reduced. Of course, readability of the information might be somewhat impaired.

The View menu Add Blank Lines and Remove Blank Lines options control the insertion or elimination of blank lines respectively. These action-bar pull-down menu options are available from most Fault Analyzer displays. (For information about Fault Analyzer action-bar pull-down menus in general, see "Action-bar pull-down menus" on page 61.)

The default setting is to add blank lines.

Only the reverse of the currently active option is available from the View menu. That is, if Add Blank Lines is in effect, then only Remove Blank Lines is available for selection and vice versa.

---

## **Adding or removing help text**

Some displays provide assistance to the inexperienced user by, for example, explaining options available or elaborating on the way information was retrieved.

The View menu Add Help Text and Remove Help Text options control the inclusion or exclusion of such help information respectively. These action-bar pull-down menu options are available from most Fault Analyzer displays (for information about Fault Analyzer action-bar pull-down menus in general, see "Action-bar pull-down menus" on page 61).

Help text in displays is enclosed in braces ({}).

The default setting is to add help text.

Only the reverse of the currently active option is available from the View menu. That is, if Add Help Text is in effect, then only Remove Help Text is available for selection and vice versa.

---

## **Setting preferred formatting width**

The Fault Analyzer ISPF interface permits the user to select a preferred formatting width. This width is the width that Fault Analyzer should use whenever possible during formatting of information for displays.

## Setting preferred formatting width

The preferred formatting width is not synonymous with the actual width of Fault Analyzer displays. It only affects the width of some displayed information, for example, text paragraphs. Other information is static by design and is not affected by the formatting width.

The Preferred Formatting Width display is shown when the View menu Preferred Formatting Width option is selected (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61).

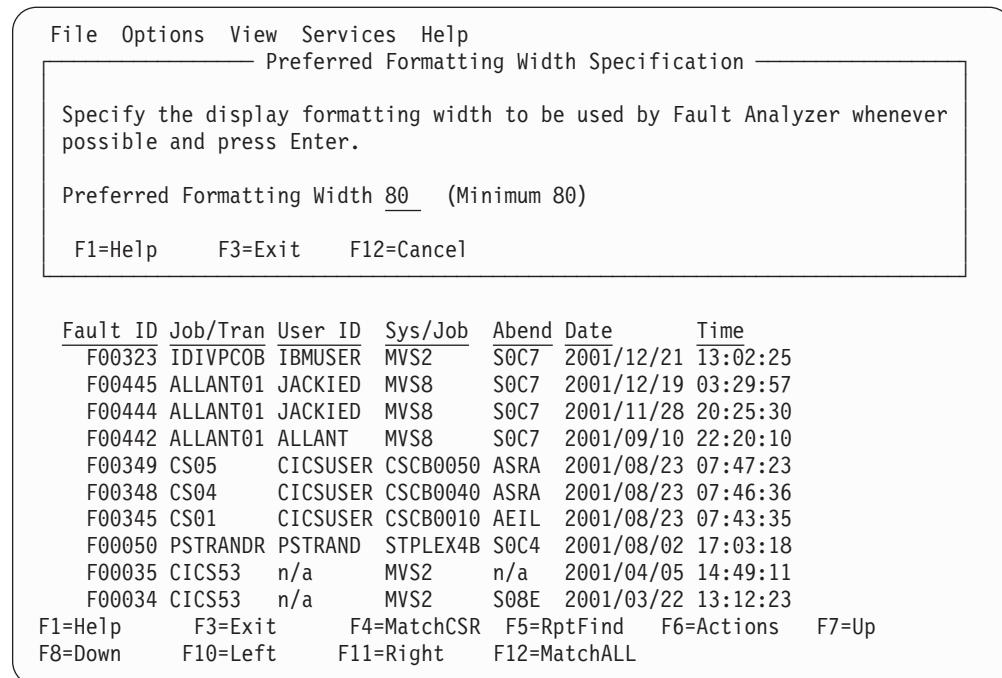


Figure 23. Sample Preferred Formatting Width display

The minimum accepted width is 80 characters, which is also the default. Although a value of 999 can be specified, the maximum width is at all times limited by the actual display width.

After typing the desired formatting width, press the Enter key to return to the previous display using the specified formatting width.

If the formatting width exceeds the physical width of your screen, then you can use the scroll commands, LEFT (PF10) and RIGHT (PF11) to view the entire display.

To return to the previous display without changing the formatting width, enter the EXIT (PF3) or CANCEL (PF12) command.

## Displaying user-selected message or abend code explanations

The LOOKUP command can be issued from the interactive fault history file display or the interactive fault reanalysis report. The command can be used to display explanations for user-selected message IDs or abend codes. It can also be used to display other information, such as DB2 SQLCODEs or VSAM feedback codes. (For the command syntax, see “LOOKUP” on page 69.)

For example, if entering the command:

## Displaying user-selected message or abend code explanations

LOOKUP IEC141I

a panel containing the message explanation is displayed as shown in Figure 24.

```
Message IEC141I Explanation                                Line 1 Col 1 80
Command ===> _____                                     Scroll ===> CSR

IEC141I
IEC141I 013-rc,mod,jjj,sss, ddname[-#] [,dev,volser, dsname]

Explanation: An error occurred during the processing of an OPEN macro.
System completion code 013, with the return code, accompanies this
message.

In the message text:

rc
      The return code.

mod
      The name of the module in which the error was detected.

jjj
      The job name.

F1=Help   F3=Exit   F7=Up   F8=Down   F12=Cancel
```

Figure 24. Sample Message ID Look-Up display

If the LOOKUP command is entered without any parameters, a display from which a message ID, abend code, or other available item of information can be specified is shown:

```
Lookup Search and Browse                                Line 1 Col 1 80
Command ===> _____                                     Scroll ===> CSR

Either search for abend codes, messages, and miscellaneous information by
typing a pattern, or browse such information using the expand/collapse browser
below.

Search. . . . . . . . . : _____

+ Abend Codes
+ Messages
+ Miscellaneous Information

F1=Help   F3=Exit   F7=Up   F8=Down   F12=Cancel
```

Figure 25. Sample Lookup Search and Browse display

## Displaying user-selected message or abend code explanations

The Lookup Search and Browse display allows you to either specify an argument in the Search field (same syntax as shown in "LOOKUP" on page 69), or you can navigate to the desired information by using the expand (+) and collapse (-) point-and-shoot fields.

By placing the cursor on a + (expand) point-and-shoot field, the available subcategories are shown. For example, if placing the cursor on the + sign next to Abend Codes in Figure 25 on page 78 and pressing the Enter key, the following expanded information becomes available:

- Abend Codes
  - + IMS User Abend Codes
  - + Language Environment User Abend Codes
  - + CICS User Abend Codes
  - + MVS Abend Codes
  - + Messages
  - + Miscellaneous Information

If next IMS User Abend Codes is expanded, a list of selectable IMS user abend code explanations is provided:

- Abend Codes
  - IMS User Abend Codes
    - U0001**
    - U0002**
    - U0005**
    - U0008**
    - U0009**
    - .
    - .
    - U3469**
    - U3498**
    - U3499**
    - U3999**
    - U4095**
  - + Language Environment User Abend Codes
  - + CICS User Abend Codes
  - + MVS Abend Codes
  - + Messages
  - + Miscellaneous Information

Only information that is provided by Fault Analyzer, or specified in member IDIHUSR.M of the IDI.SIDISAM1 data set, can be entered.

---

## Copying interactive displays to a file

The COPY command can be used to write a copy of the current display (which includes the entire scrollable area) to a sequential data set or a member in a partitioned data set. (For the command syntax, see "COPY" on page 65.)

For example, if entering the command:

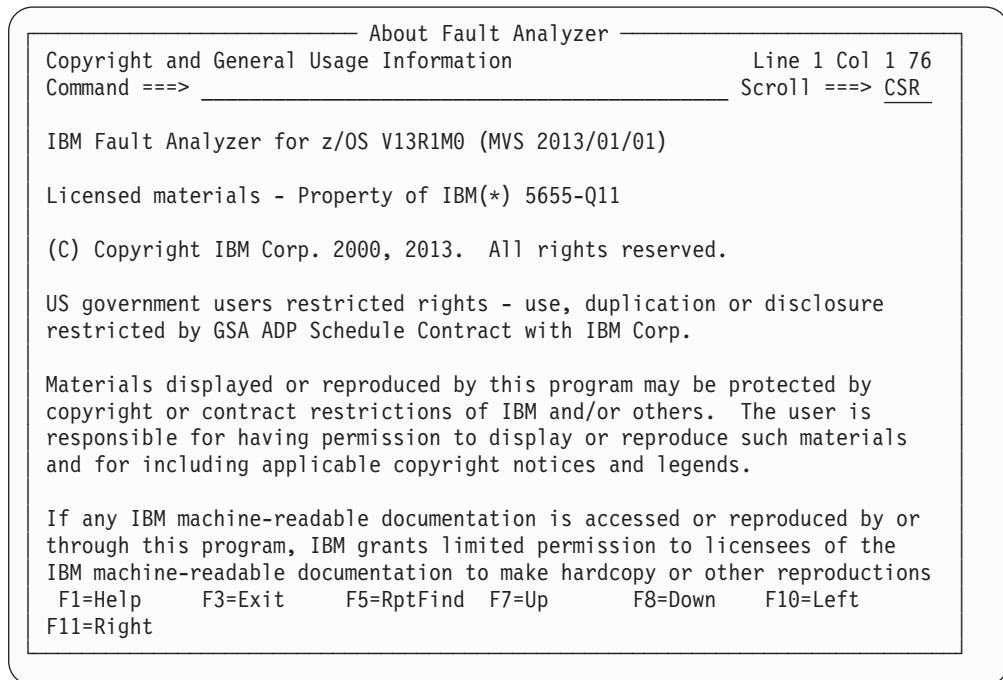
```
COPY PRINT(A)
```

the current display is written to member A in the partitioned data set '*prefix.PRINT*', where *prefix* is the current TSO prefix.

If the COPY command is entered without any parameters, a pop-up panel from which the data set name can be specified is displayed.

## **Displaying Fault Analyzer copyright and general usage information**

Fault Analyzer copyright and general usage information is available by selecting the Help menu About Fault Analyzer option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This option brings up the About Fault Analyzer display as the example shown in Figure 26.



*Figure 26. Sample About Fault Analyzer display*

Also available from this display is the currently installed version, release, and maintenance level of Fault Analyzer, including the last ++APAR applied.

## **Deleting history file entries**

You delete by entering the line command D against the fault entry in the Fault Entry List display.

You can also delete a range of fault history entries, using two DD range markers. When you delete a range of a matched set of entries, you only delete the entries displayed on the screen. You do not delete any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

**Note:** Deletion of a fault entry is not possible if the entry's lock flag is not blank, or if the specified number of days since the fault entry's creation has not yet elapsed. For more information about the lock flag, and how to change its value, see “Viewing fault entry information” on page 82.

If one or more fault entries could not be deleted due to insufficient access authority, or due to the fault entries being locked, then a display detailing the reasons is shown.

What happens when you attempt to delete depends on your “Confirm Fault Entry Deletion” option which is explained in the following.

## Changing deletion options through the Options menu

To change the deletion option, first select the Options menu Fault Analyzer Preferences option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens the Fault Analyzer Preferences display as illustrated in Figure 27.

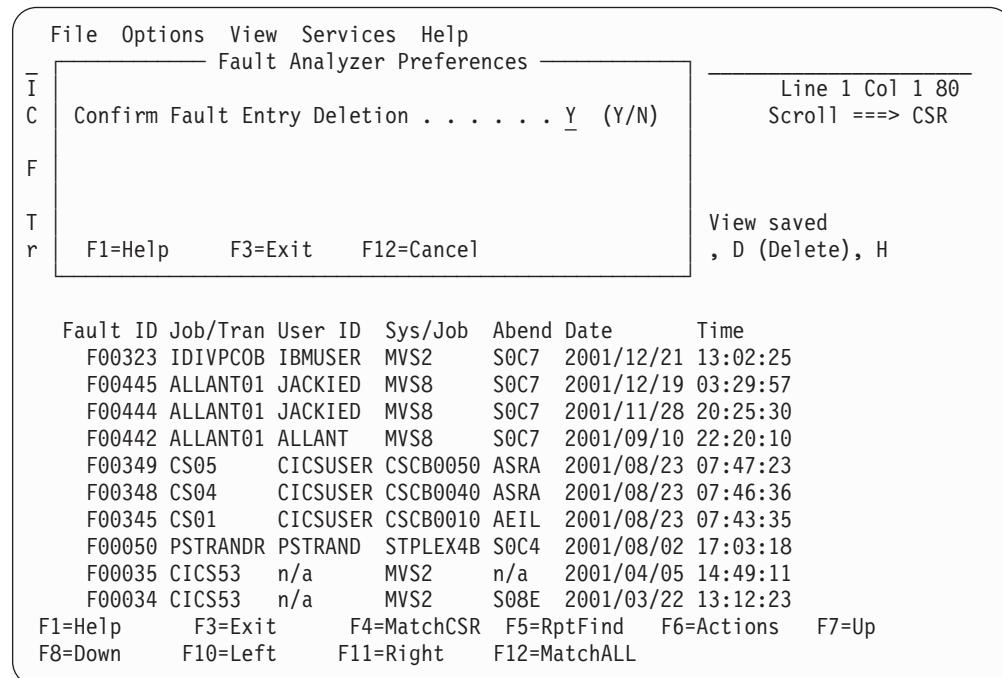


Figure 27. Sample Fault Analyzer Preferences display

The option you can set is:

### Confirm Fault Entry Deletion

If this field is set to “Y”, then the Confirm Fault Entry Deletion display is shown each time when one or more fault entries are attempted deleted. If set to “N”, then the confirmation display is not shown, which is faster. You can also change this option from the confirmation display.

## Deleting when the confirmation display is shown

If the “Confirm Fault Entry Deletion” option is set to Y, then the Confirm Fault Entry Deletion display is shown.

## Deleting history file entries

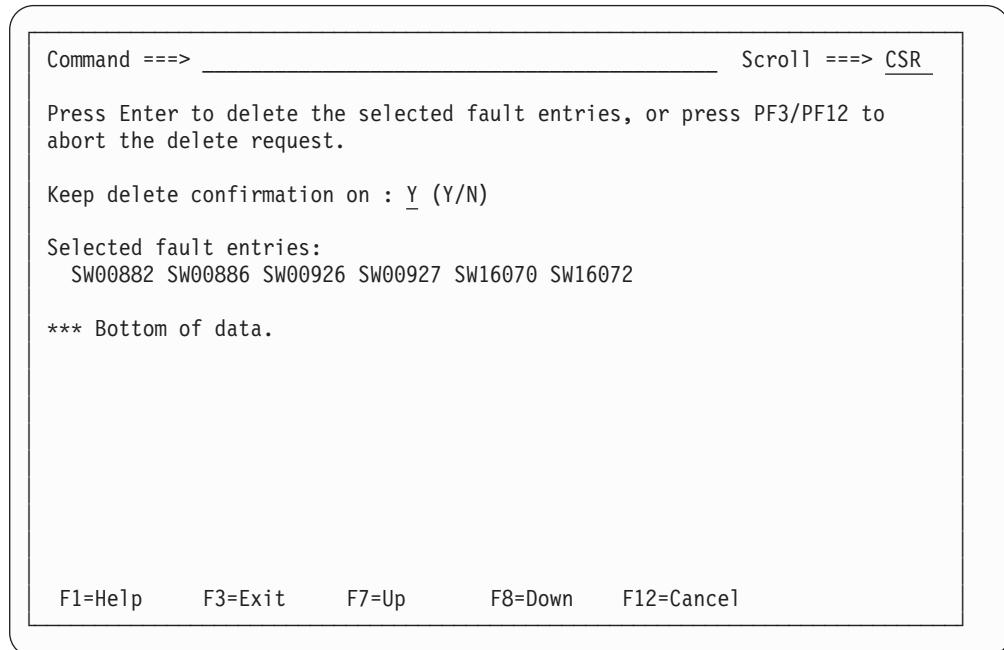


Figure 28. Sample Confirm Fault Entry Deletion display

If for subsequent deletes, you do not want to see the confirmation display, then you change the "Keep delete confirmation on" option to N. If at some later stage you want to see the confirmation display, then you can change the option ("Confirm Fault Entry Deletion") from the "Options" menu.

## Deleting when the confirmation display is not shown

If the "Confirm Fault Entry Deletion" option is set to N, then the confirmation display is not shown. Instead, the deletion proceeds.

## Bulk deletions

To delete many history file entries, enter DD on the first fault history entry, and DD on the last fault history entry. This action sets up the deletion of these two entries, and all the entries in between. You can use a match to group together the entries you want to delete.

If the "Confirm Fault Entry Deletion" option is set to Y, then the confirmation window is displayed, although you can suppress the display for following files in the bulk deletion by setting the "Keep delete confirmation on" option to N.

## Viewing fault entry information

The history file is displayed with one line of information per entry. Therefore, not all information pertaining to a fault can be displayed simultaneously.

To view more information, enter the ? line command against the entry on the Fault Entry List display. From within the interactive reanalysis report, the same can be accomplished by either issuing the INFO command (see "INFO" on page 69) or by selecting "Fault Entry Information" from the action-bar "File" pull-down menu (see "Action-bar pull-down menus" on page 61).

An example of the fault entry information display follows:

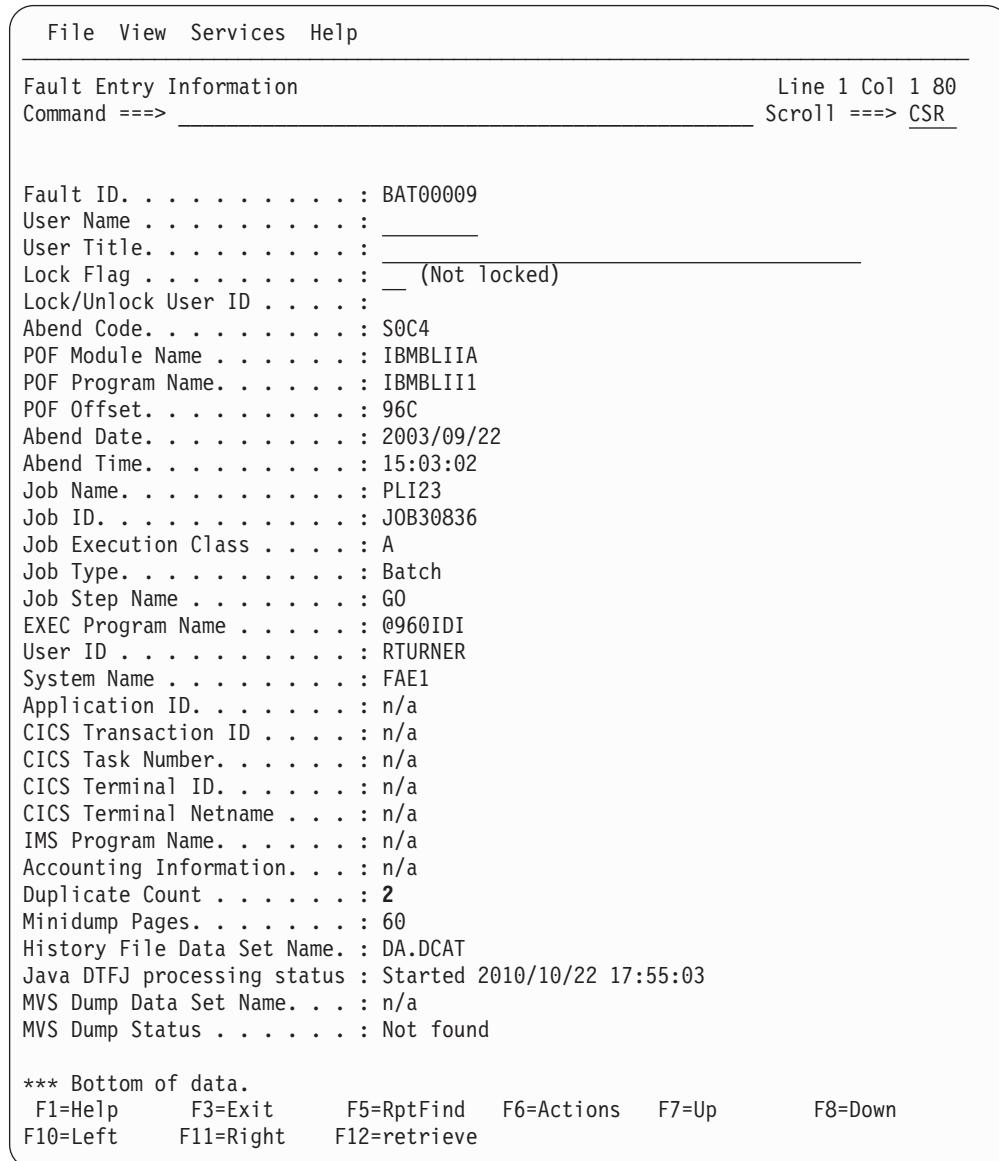


Figure 29. Sample Fault Entry Information display

Depending on screen size, it might be necessary to scroll down to see all information.

The display includes the following fields:

#### Fault ID

The ID of the fault entry selected.

#### User Name

A user-maintained input/output field that might contain, for example, the name, or ID, of the person to whom the fault is assigned. To change the contents of this field, remember to press the Enter key prior to returning with PF3. The initial content of this field can be provided via IDISNAP.

#### User Title

A user-maintained input/output field that is intended to contain a short description of the fault. If this field is used, then it is displayed at the top of any batch reanalysis report, and at the top of the first display of the

## Viewing fault entry information

interactive reanalysis report. To change the contents of this field, remember to press the Enter key prior to returning with PF3.

### Lock Flag

A user-maintained input/output field that provides a mechanism to prevent accidental deletion of the fault entry.

If this field contains a numeric value 0 - 99, then fault entry expiration control is active. The specified value is the number of days that the fault entry remains locked, following its creation. The time of the day is not considered. If, for example, a value of 1 is specified when the fault entry is created, it is unlocked at midnight on the same day. It follows that a value of 0 means that the fault entry is not locked. If fault entry expiration control is active, but has not yet expired, or if setting this flag to any other non-blank character, then the following is not performed against the fault entry:

- IDIUTIL DELETE processing (for details, see “DELETE control statement” on page 339).

**Note:** The default is to not delete the fault entry, but this default can be overridden by using an IDIUTIL Delete user exit (for details, see “IDIUTIL Delete user exit” on page 404).

- Deletion from the ISPF interface Fault Entry List display, using the 'D' or 'DD' line commands (for details, see “Deleting history file entries” on page 80).
- Automatic deletion due to the maximum number of fault entries for a history file having been reached, and the fault entry which is locked is the oldest fault entry in the history file. In this case, the search for a fault entry to delete continues in chronological order until the oldest fault entry that is not locked has been located.

By default, the lock flag is set to a blank, which does not prevent deletion of the fault entry.

Any printable character can be entered for this field:

- If a non-printable character is entered, then it is changed to a '/'.
- If a lowercase character is entered, then it is translated to uppercase.

This flag can also be modified by any user exit by changing the value in the ENV.LOCK\_FLAG field (for details, see “ENV - Common exit environment information” on page 513).

The current lock status is shown following the lock flag, and is updated if pressing the Enter key after changing its value.

### Lock/Unlock User ID

Identifies the user ID who last changed the Lock Flag setting.

### Abend Code

The initial (if more than one) abend code.

For a fault entry created using IDISNAP, the abend code is shown as "SNAP".

### POF Module Name

The point-of-failure module name.

### POF Program Name

The point-of-failure program name.

**POF Offset**

The point-of-failure offset.

**Abend Date**

The date when the abend occurred.

**Abend Time**

The time when the abend occurred.

**Job Name**

The name of the batch job, started task, or TSO user ID that caused the entry to be written.

**Job ID**

The JES ID of the abending job.

**Job Execution Class**

The JES execution class of the abending job.

**Job Type**

The abending job type.

**Job Step Name**

The name of the job step that caused the entry to be written.

**EXEC Program Name**

The program name that is specified on the JCL EXEC statement of the abending job step.

**User ID**

The user ID associated with the abending job.

**System Name**

The system name on which the abend occurred.

**CICS Transaction ID**

If a CICS transaction fault, the ID of the CICS transaction that caused the entry to be written.

**CICS Task Number**

If a CICS transaction fault, the task number that caused the entry to be written.

**CICS Terminal ID**

If a CICS transaction fault, the CICS terminal ID.

**CICS Terminal Netname**

If a CICS transaction fault, the CICS terminal netname.

**IMS Program Name**

IMS program name.

**Accounting Information**

Job accounting information.

**Duplicate Count**

The number of duplicate faults that have occurred, using the same fault history file, since the recording of this fault.

**Note:** This value might not always include all duplicate faults that have occurred as it is maintained in the volatile history file cache only for performance reasons.

If a non-zero value is displayed, and duplicate details are available for the fault, then the value becomes a point-and-shoot field, which can be

## Viewing fault entry information

selected by placing the cursor on it, and pressing Enter. This selection results in the Fault Entry Duplicate History display being shown (for details, see “Viewing the fault entry duplicate history” on page 87).

### Minidump Pages

The number of minidump pages written to the history file for this fault.

### History File Data Set Name

The name of the history file data set containing the viewed fault entry.

### Java DTFJ Processing Status

For a fault entry for which Java activity was detected, this field shows the status of the asynchronous Java DTFJ processing as one of the following:

- SVC dump not available
- Pending
- Started *date time*
- Finished, elapsed seconds *seconds*

### MVS Dump Data Set Name

The MVS dump (SYSMDUMP or SVC dump) data set name.

The dump data set name is not displayed if the status is Suppressed, or if the dump data set name is not available.

### MVS Dump Status

The MVS dump data set status.

The MVS dump data set status field shows you information about the dump that can be associated with the current fault. Possible values of this field are:

#### Available

A SYSMDUMP data set name was recorded from the Job Step at the time of the fault.

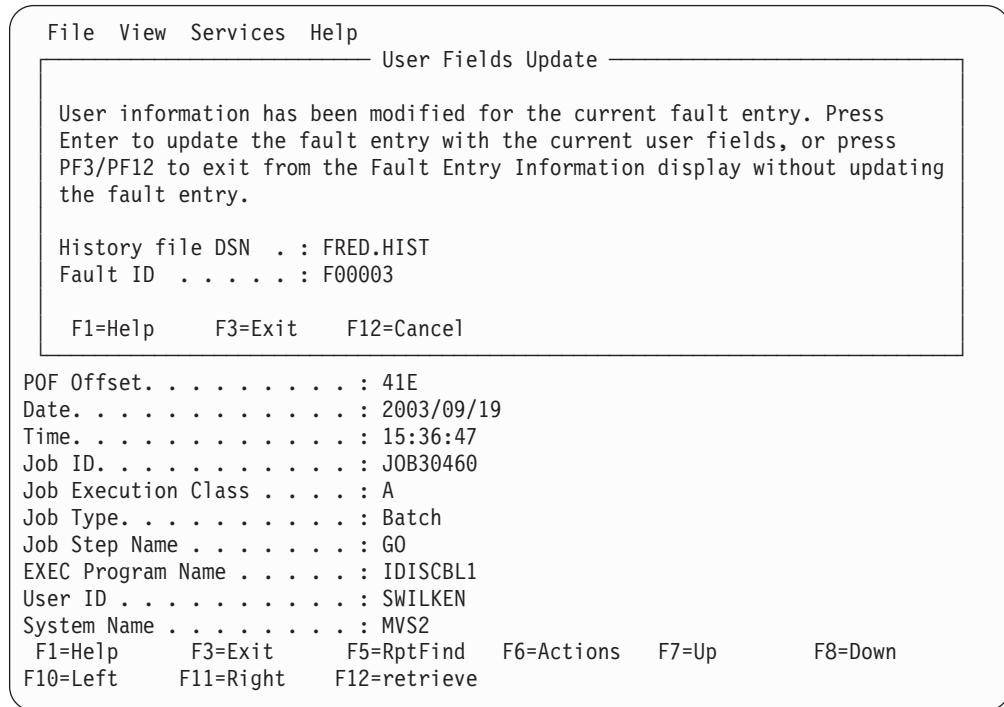
#### Not found

No SYSMDUMP data set name was recorded at the time of the fault or the recorded data set name is no longer cataloged.

#### Suppressed

The analysis provided by Fault Analyzer provides enough information about the cause of the fault, so a SYSMDUMP was not written. (If you request the RetainDump(ALL) option, Fault Analyzer does not suppress the dump, even if the analysis is deemed complete.)

When you exit from the Fault Entry Information display, after you have changed either of the user-managed fields, a prompt is displayed to confirm that you want to update the fault entry with the new information. An example of the prompt follows:



*Figure 30. Sample User Fields Update prompt*

To update the fault entry, press Enter. Otherwise, to prevent updating the fault entry, press PF3 or PF12.

## **Viewing the fault entry duplicate history**

The Fault Entry Duplicate History display provides information about additional occurrences of the fault, within the NoDup option time limit in effect for the fault type and subject to the defined duplicate criteria.

The display can be invoked by either:

- Entering the H line command against a fault entry from the Fault Entry List display.
  - Placing the cursor on the Fault Entry List display Dups column value for a fault entry, when this entry is presented as a point-and-shoot field, and pressing Enter.
  - Issuing the DUPS primary command from within the interactive reanalysis report.
  - Placing the cursor on the Fault Entry Information display Duplicate Count value, when this value is presented as a point-and-shoot field, and pressing Enter.

An example of a Fault Entry Duplicate History display follows:

## Viewing the fault entry duplicate history

The screenshot shows a terminal window with the following content:

File View Services Help

Fault Entry Duplicate History  
Command ==> \_\_\_\_\_ End of data  
\_\_\_\_\_  
Scroll ==> CSR

Most recent duplicate  
occurred. . . . . : 2005/06/28 15:22:13  
Initial abend occurred. . . : 2005/06/28 15:20:55  
Total duplicate count . . . : 4

Duplicate details in reverse chronological order:

Date	Time	Jobname	Job ID	System	Dup Type	User ID	Stepname
2005/06/28	15:22:13	CICS5FA1	JOB48697	FAE1	Normal	SIMCOC2	CICS5FA1
2005/06/28	15:22:02	CICS5FA1	JOB48697	FAE1	Normal	SIMCOC2	CICS5FA1

Date	Time	Jobname	Job ID	System	Dup Type
2005/06/28	15:21:08	CICS5FA1	JOB48697	FAE1	Fast
User ID	Term ID	Count			
SIMCOC2	1265	2			

\*\*\* Bottom of data.  
F1=Help F3=Exit F5=RptFind F7=Up F8=Down F10=Left  
F11=Right

Figure 31. Sample Fault Entry Duplicate History display

The following information is provided:

### Most recent duplicate occurred

The local date and time when the most recent duplicate of this fault occurred. The formatting of the date and time is subject to the Locale option in effect.

### Initial abend occurred

The local date and time when the initial fault that caused the entry to be written occurred. The formatting of the date and time is subject to the Locale option in effect.

### Total duplicate count

The total number of duplicates that have been registered against this fault.

### Duplicate details in reverse chronological order

Details of the duplicate occurrences.

Depending on whether the duplicates resulted from NoDup(Normal) or NoDup(CICSFast) processing, the information that is provided differs:

- NoDup(Normal) duplicates

For NoDup(Normal) duplicates, the information that is provided for each instance is comprised of:

**Date** The date when the duplicate occurred. The formatting of the date is subject to the Locale option in effect.

**Time** The time when the duplicate occurred. The formatting of the time is subject to the Locale option in effect.

### Jobname

The name of the job that caused the duplicate fault.

### Job ID

The JES job ID of the job that caused the duplicate fault.

## Viewing the fault entry duplicate history

### System

The name of the system on which the job that caused the duplicate fault was executing.

### Dup Type

The type of duplicate (always "Normal" for NoDup(Normal) duplicates).

### User ID

The user ID that is associated with the job that caused the duplicate fault.

### Stepname

The name of the job step that caused the duplicate fault.

Multiple consecutive NoDup(Normal) occurrences are grouped under a common heading.

- NoDup(CICSFast) duplicates

For NoDup(CICSFast) duplicates, the information that is provided consists of:

- A common section, with details that are applicable to all duplicate faults (one or more) that occurred within the recording interval:

**Date** The date when the first duplicate occurred within the recording interval. The formatting of the date is subject to the Locale option in effect.

**Time** The time when the first duplicate occurred within the recording interval. The formatting of the time is subject to the Locale option in effect.

### Jobname

The name of the job that caused the duplicate faults.

### Job ID

The JES job ID of the CICS region that caused the duplicate faults.

### System

The name of the system on which the CICS region that caused the duplicate faults was executing.

### Dup Type

The type of duplicates (always "Fast" for NoDup(CICSFast) duplicates).

- Detailed information of up to 10 unique faults that occurred within the recording interval:

### User ID

The CICS user ID that caused the duplicate faults.

### Terminal ID

The CICS terminal ID that caused the duplicate faults.

**Count** The total number of occurrences of this unique combination of CICS user ID and terminal ID that occurred within the recording interval.

**Note:** If the sum total of all "Count" values does not equal the total number of duplicates that occurred within the recording interval, then a note is provided with information about the number of duplicates for which details are not available.

## Viewing the fault entry duplicate history

This situation can happen if duplicate faults for more than 10 unique combinations of CICS user IDs and terminal IDs occurred during the recording interval.

**Note:** If the sum total of all duplicate faults shown does not equal the total duplicate count for the fault entry, then a note is provided with information about the number of duplicates for which details are not available.

This situation can happen if more duplicate faults have occurred than can be recorded with details in the fault entry.

## Copying history file entries

You copy by entering the line command C against the fault entry in the Fault Entry List display.

You can also copy a range of fault history entries, using two CC range markers. When you copy a range of a matched set of entries, you only copy the entries displayed on the screen. You do not copy any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for copy with the C or CC commands is presented. Included are also any fault entries that are selected for move with the M or MM commands. The display provides a field for specification of a history file to which the fault entries should be copied. This field is initialized with the name of the history file last specified. The following is an example of the Specify Move/Copy Options display that is shown after pressing the Enter key.

Specify Move/Copy Options	Line 1 Col 1 76			
Command ==> _____	Scroll ==> CSR			
Verify or change move/copy options for the selected fault entries and press Enter, or press PF3/PF12 to abort the move/copy request.				
Move/Copy Options:				
Destination History File: 'IDI.COPY.HIST'				
Selected Fault Entries:				
BAT14648 BAT14649 BAT14650 BAT14651 BAT14652 BAT14653 BAT14654 BAT14655 BAT14656 BAT14657 BAT14658 BAT14659 BAT14660 BAT14661 BAT14662 BAT14663 BAT14664 BAT14665 BAT14666 BAT14667 BAT14668 BAT14669 BAT14670 BAT14671 BAT14672				
*** Bottom of data.				
F1=Help	F3=Exit	F7=Up	F8=Down	F12=Cancel

Figure 32. Sample Specify Move/Copy Options display

The destination history file that is specified is checked for UPDATE access authorization prior to commencing the move/copy processing. If the specified

history file does not exist, then the New History File Allocation display is shown (for details, see “New history file allocation” on page 55).

The original fault ID is preserved, if possible. However, if an identical fault ID already exists in the destination history file, then the fault number is incremented to the next available fault ID.

If the copy of one or more fault entries was not successful, then a display detailing the reasons is shown.

The TSO user's access authorization level required to copy a fault entry is READ.

---

## Moving history file entries

You move by entering the line command M against the fault entry in the Fault Entry List display.

You can also move a range of fault history entries, using two MM range markers. When you move a range of a matched set of entries, you only move the entries displayed on the screen. You do not move any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for move with the M or MM commands is presented. Included are also any fault entries that are selected for copy with the C or CC commands. The display provides a field for specification of a history file to which the fault entries should be moved. This field is initialized with the name of the history file last specified. See “Copying history file entries” on page 90 for an example of the Specify Move/Copy Options display that is shown after pressing the Enter key.

The destination history file that is specified is checked for UPDATE access authorization prior to commencing the move/copy processing. If the specified history file does not exist, then the New History File Allocation display is shown (for details, see “New history file allocation” on page 55).

The original fault ID is preserved, if possible. However, if an identical fault ID already exists in the destination history file, then the fault number is incremented to the next available fault ID.

If the move of one or more fault entries was not successful, then a display detailing the reasons is shown.

A locked fault entry cannot be moved. If you need to move a locked fault entry, then first issue the '?' line command and unlock the fault entry from the Fault Entry Information display by changing the Lock Flag value to blanks. If required, lock the moved fault entry again by changing the Lock Flag value to a non-blank value. For details of the values that are permitted for the Lock Flag, see “Viewing fault entry information” on page 82.

The TSO user's access authorization level required to move a fault entry is ALTER.

---

## Transmitting history file entries

You XMIT by entering the line command X against the fault entry in the Fault Entry List display.

## Transmitting history file entries

You can also XMIT a range of fault history entries, using two XX range markers. When you XMIT a range of a matched set of entries, you only XMIT the entries displayed on the screen. You do not XMIT any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for XMIT with the X or XX commands is presented. The display provides fields for specification of the destination node and user ID to which the fault entries should be transmitted, as well as selection of the transmitted data set type as either sequential (SEQ) or partitioned (PDS). These fields are initialized with the values last specified. The following is an example of the Specify XMIT Options display that is shown after pressing the Enter key.

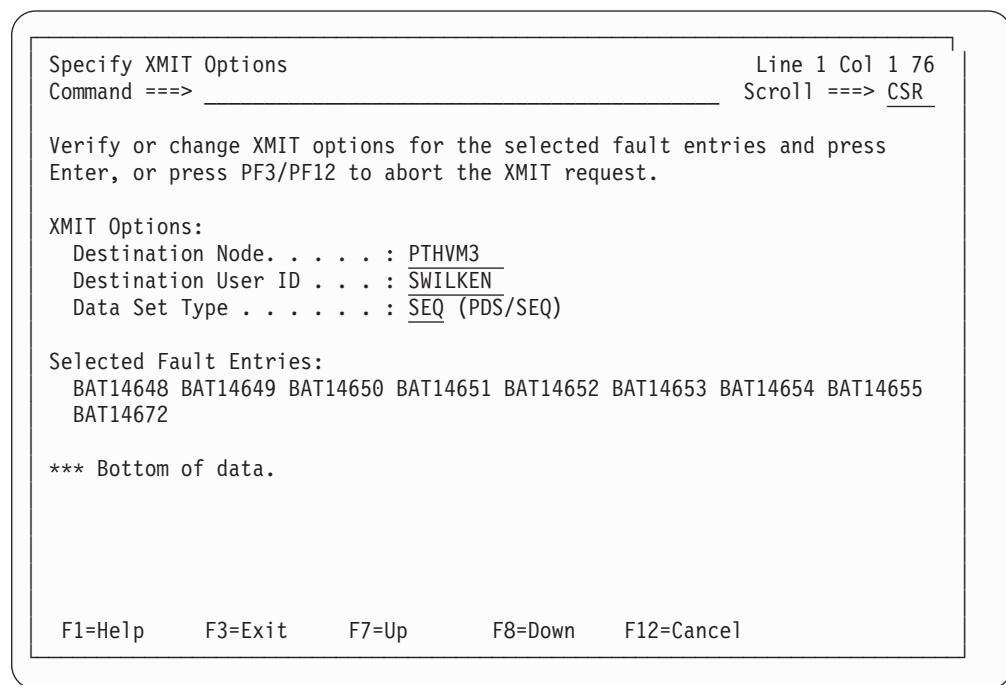


Figure 33. Sample Specify XMIT Options display

Each selected fault entry is transmitted separately to the specified destination.

If the XMIT of one or more fault entries was not successful, then a display detailing the reasons is shown.

The TSO user's access authorization level required to XMIT a fault entry is READ.

## Security considerations

The best way to manage the access to history file fault entries is by using the security server XFACILIT resource class as described in “Using the XFACILIT resource class for history file fault entries” on page 274.

Alternatively, the use of Views (see “Using views” on page 37) in conjunction with data set profile security might be considered.

---

## Specifying 64-bit addresses

The following applies to display input fields and command parameters where 64-bit addresses are supported.

A 64-bit enabled address field or command parameter can be specified by up to 16 hexadecimal digits, and might optionally include an underscore between bits 31 and 32. If an underscore is specified, then the value on the right hand side of the underscore is automatically padded with leading zeroes to form 8 digits. For example, the following are all considered identical address specifications:

```
100100000  
1_100000  
1_00100000  
0000000100100000  
00000001_00100000
```

## **Transmitting history file entries**

---

## Chapter 4. Performing batch reanalysis

Batch reanalysis might be used when a user wants to either rerun analysis on a number of faults, or would prefer the reanalysis to run in batch rather than hold up the TSO session. The batch report is directed to the SYSPRINT DD statement of the batch job step.

The batch reanalysis report looks the same as the real-time analysis report. For details, see Chapter 9, “The Fault Analyzer report,” on page 197.

---

### Batch reanalysis options

Many of the general options specified for an installation are also applicable to batch reanalysis. See Chapter 33, “Options,” on page 451 for information about all available options and the different ways in which they can be specified.

Some of the options that you might want to consider using to control the batch reanalysis report are:

**Detail** Specify this option if you want to adjust the level of detail given in the batch reanalysis report. See “Detail” on page 465 for more detail.

**PrintInactiveCOBOL**

The PrintInactiveCOBOL option can be used to request that storage for inactive COBOL programs (programs that are not in the current save-area chain) is included in the reanalysis report.

To specify batch reanalysis options that apply to your batch jobs only, first select Batch Reanalysis Options from the Fault Entry List display Options menu (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens the Batch Reanalysis Options display as shown in Figure 34 on page 96.

## Batch reanalysis options

File View Services Help

Batch Reanalysis Options Line 1 Col 1 80  
Command ==> Scroll ==> CSR

Press PF3 to save options or PF12 to cancel.

General Options:  
Options line for batch reanalysis. . . . . : Detail(Long)

Redisplay this panel before each reanalysis. . . : N (Y/N)

Display panel to edit generated JCL . . . . . : N (Y/N)

Job card style. . . . . : P (P=Parameters, S=Statements)

Job Card Parameters:  
Job name suffix . . . . . : A (A-Z, 0-9, @, #, or \$)  
Job class . . . . . : A (A-Z or 0-9)  
Job notify. . . . . : Y (Y/N)  
Job time minutes. . . . . : 10 (0-99)  
Message class . . . . . : X (A-Z or 0-9)  
Region megabytes. . . . . : 0 (0-2047)  
Accounting info . . . . . :

Reanalysis Report:  
Destination . . . . . :

Reanalysis Options Data Set Control:  
Options data set name . . . :  
Options member name . . . : \_\_\_\_\_ (If PDS or PDSE)  
Use this data set during reanalysis. . . . . : N (Y/N)  
Edit the options data set before reanalysis . . . . . : N (Y/N)

\*\*\* Bottom of data.  
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
F10=Left F11=Right F12=Cancel

Figure 34. Sample Batch Reanalysis Options display

The following can be specified using this display:

### Options line for batch reanalysis

Options that apply to all batch reanalysis jobs that you submit can be specified here. These options, which are used in the PARM field of the generated batch reanalysis job, take precedence over any options that are specified through an options file (see "Options data set name" below).

The option Detail(Long) is shown as an example on the options line in Figure 34.

If you need to specify more options than fit on this line, then use one of the options "Display panel to edit generated JCL" or "Edit the options data set before reanalysis" instead (both are explained in the following).

Options that are specified on the options line are saved in the user profile.

### Redisplay this panel before each reanalysis

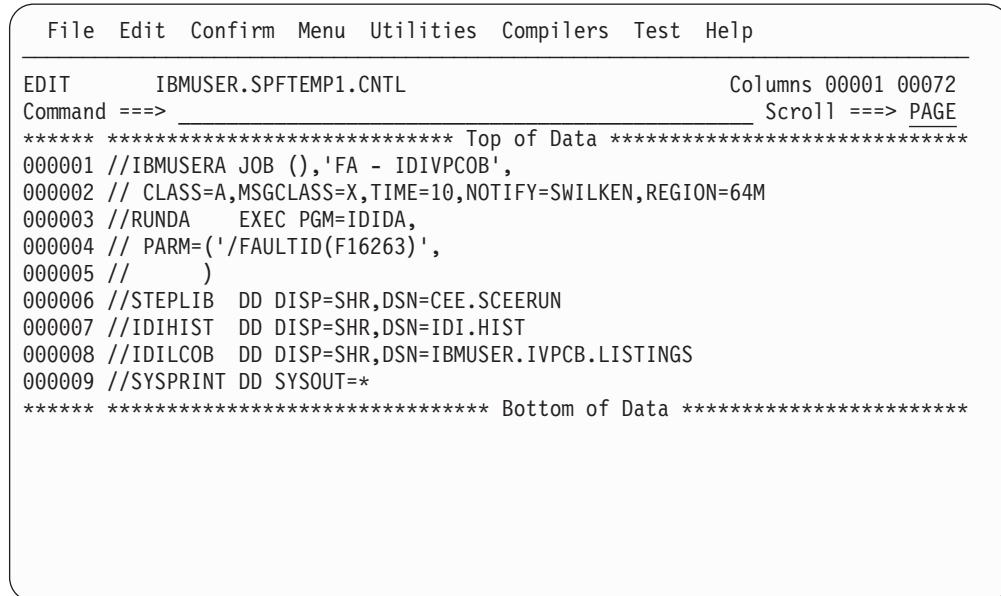
If this option is set to Y, then the Batch Reanalysis Options display is shown each time a batch reanalysis is requested, prior to generating the JCL stream.

Make your changes. Then press PF3 (to continue with the current options) or PF12 (to undo any changes made). What happens next depends on the “Display panel to edit generated JCL” option below.

If this option is set to N, then the Batch Reanalysis Options display is not shown.

### Display panel to edit generated JCL

If this option is set to Y, then you are presented with an ISPF EDIT display screen of the JCL stream generated by Fault Analyzer as the example shown in Figure 35.



The screenshot shows an ISPF EDIT window with the following details:

- File Edit Confirm Menu Utilities Compilers Test Help** (menu bar)
- EDIT IBMUSER.SPFTEMP1.CNTL Columns 00001 00072** (top left)
- Command ==> \_\_\_\_\_ Scroll ==> PAGE** (top right)
- \*\*\*\*\* \*\*\*\*\* Top of Data \*\*\*\*\*** (text indicator)
- 000001 //IBMUSERA JOB (),'FA - IDIVPCOB',  
000002 // CLASS=A,MSGCLASS=X,TIME=10,NOTIFY=SWILKEN,REGION=64M  
000003 //RUNDA EXEC PGM=IDIDA,  
000004 // PARM=('/FAULTID(F16263)',  
000005 // )  
000006 //STEPLIB DD DISP=SHR,DSN=CEE.SCEERUN  
000007 //IDIHIST DD DISP=SHR,DSN=IDI.HIST  
000008 //IDILCOB DD DISP=SHR,DSN=IBMUSER.IVPCB.LISTINGS  
000009 //SYSPRINT DD SYSOUT=\*** (JCL code)
- \*\*\*\*\* \*\*\*\*\* Bottom of Data \*\*\*\*\*** (text indicator)

Figure 35. Sample batch reanalysis JCL stream EDIT

Make your changes. Then issue the SUBMIT command to execute the job. Then enter the EXIT (PF3) or CANCEL (PF12) command to return to the Fault Entry List display.

For more information about the use of this option, see “Data sets used for batch reanalysis” on page 100.

If this option is set to N, then the generated JCL stream is submitted automatically without first displaying the JCL EDIT screen.

### Job card style

A single character (P or S) that controls the style of job card specification that follows:

- If P is specified (the default), then a Job Card Parameters section, as shown in Figure 34 on page 96 follows the General Options section after the Enter key is pressed.
- If S is specified, then a Job Card Statements section follows the General Options section after the Enter key is pressed.

This would alter the display from that shown in Figure 34 on page 96 to the following:

## Batch reanalysis options

File View Services Help

Batch Reanalysis Options Line 1 Col 1 80  
Command ==> Scroll ==> CSR

Press PF3 to save options or PF12 to cancel.

General Options:

Options line for batch reanalysis. . . . . : \_\_\_\_\_

Redisplay this panel before each reanalysis. . . : N (Y/N)

Display panel to edit generated JCL . . . . . : N (Y/N)

Job card style. . . . . : S (P=Parameters, S=Statements)

Job Card Statements:

====> \_\_\_\_\_  
====> \_\_\_\_\_  
====> \_\_\_\_\_  
====> \_\_\_\_\_

Reanalysis Report:

Destination . . . . . : \_\_\_\_\_

Reanalysis Options Data Set Control:

Options data set name . . . : \_\_\_\_\_  
Options member name . . . : \_\_\_\_\_ (If PDS or PDSE)  
Use this data set during reanalysis. . . . . : N (Y/N)  
Edit the options data set before reanalysis . . . . : N (Y/N)

\*\*\* Bottom of data.  
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
F10=Left F11=Right F12=Cancel

Figure 36. Sample Batch Reanalysis Options display

Whichever style of job card specification is selected with this option determines the method used when generating the job card for all batch reanalysis jobs.

### Job name suffix

This suffix is the character that is appended to your user ID to form the job name that is used for the generated batch reanalysis JCL stream. The default is A.

### Job class

This job class is the job class that is used on the CLASS parameter of the generated JOB card. The default is A.

### Job notify

If this field is set to Y, then a NOTIFY=*userid* parameter is added to the generated JOB card. If it is set to N, then no NOTIFY parameter is added. The default is Y.

### Job time minutes

This value is the number of minutes that are used on the TIME parameter of the JOB card. The valid range is 1 - 30. The default is 10.

### Message class

This class is the message class that is used on the MSGCLASS parameter of the generated JOB card. The default is X.

**Region megabytes**

This value is the value that is used on the REGION parameter of the generated JOB card. The valid range is 0 - 2047. The default is 0.

**Accounting info**

Anything specified in this field is used as accounting information on the generated JOB card. The default is not to provide any accounting info.

**Destination**

An optional specification of the reanalysis report destination. The specified option must be correct for adding to a DEST parameter on the generated SYSPRINT DD statement used for the report.

**Options data set name**

This field can optionally specify the name of a PDS(E) data set in which a member (see "Options member name") contains Fault Analyzer options. The data set and member name are used as the IDIOPTS user options file. This data set can, for example, be used if more options than fit on the options line at the top of this display are required.

**Note:**

1. The options data set is only used if the "Use this data set during reanalysis" option is set to Y.
2. Options that are specified on the options line take precedence over options specified in this data set.

**Options member name**

This name is the member name of the data set specified in "Options data set name".

**Use this data set during reanalysis**

If this option is set to Y, then the data set and member name specified above are used by Fault Analyzer during the batch reanalysis. If it is set to N, then the data set and member name are not used.

**Edit the options data set before reanalysis**

If this field is set to Y, then an ISPF EDIT display screen of the member in the options data set specified above is presented prior to generating the batch reanalysis JCL stream. An example is shown in Figure 37 on page 100.

## Batch reanalysis options

The screenshot shows an ISPF EDIT session window. The menu bar includes File, Edit, Confirm, Menu, Utilities, Compilers, Test, and Help. The title bar displays "EDIT MY.OPTIONS(SAMPCNF) - 01.02". The command line shows "Command ==>". The status bar indicates "Columns 00001 00072" and "Scroll ==> PAGE". The main area contains the following text:

```
***** **** Top of Data ****
000001 detail(1)
***** **** Bottom of Data ****
```

At the bottom of the window, function key definitions are listed:

F1=Help	F2=Split	F3=Exit	F4=Return	F5=Rfind	F6=Rchange
F7=Up	F8=Down	F9=Swap	F10=Left	F11=Right	F12=Cancel

Figure 37. Sample options file EDIT for batch reanalysis

Change the options data set (if you wish). Then enter the EXIT command (usually mapped to PF3).

## Initiating batch reanalysis

To initiate batch reanalysis, enter **B** against the fault history entry.

Depending on your batch options, one or more displays might be shown prior to the submission of the Fault Analyzer generated JCL stream. For details, see “Batch reanalysis options” on page 95.

## Data sets used for batch reanalysis

When performing batch reanalysis through the ISPF interface, the generated JCL includes DD statements as required for any JOBLIB, STEPLIB, or Fault Analyzer compiler listing or side file data sets. DD statements are added for Fault Analyzer data sets even if they were not explicitly included in the real-time JCL, but were supplied through the DataSets option or an Analysis Control user exit. These data sets are added to the reanalysis job in an attempt to recreate the same execution environment as were used in real time.

DataSets options that are specified via the IDIOPTS user options file or the PARM field cause those data sets to be logically concatenated to the data sets from the real-time execution.

If the “Display panel to edit generated JCL” option on the Batch Reanalysis Options display is set to Y (see “Batch reanalysis options” on page 95), then it is possible to make changes to the real-time data set specifications before submitting the reanalysis job. Also, any data sets that were used in real time but do not exist in the reanalysis environment, or data sets with READ access prohibited, are identified by a comment as shown in the following example for IDILCOB:

```
//IDILCOB DD DISP=SHR,DSN=CTEST.DUMPA.LISTING.CICS.COBOL
//          DD DISP=SHR,DSN=DA.LISTING.COBOL
//** The following IDILCOB data set is unavailable:
//**          DD DISP=SHR,DSN=CTEST.DUMPA.LISTING.CICS.COBOVS
//** The following IDILCOB data set is READ protected:
//**          DD DISP=SHR,DSN=CTEST.PROTECT.LISTINGS
```

---

## Creating your own batch reanalysis job

Batch mode can also be invoked via JCL that you have created or saved from a job previously generated by Fault Analyzer and later modified if necessary. As identification of the fault to be reanalyzed you need to specify either a fault ID (using the FaultID option), or a SYSMDUMP or SVC dump data set name (using the DumpDSN option). Your job might, for example, look like this:

---

```
//RTURNERA JOB (),'FAULT ANALYZER',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*
/* Allocate a PDSE for compiler listings
/*
//ALLOC EXEC PGM=IEFBR14
//DD1   DD DISP=(,CATLG),DSN=&SYSUID..COBLIST,SPACE=(CYL,(1,1,5)),
//          DCB=(RECFM=FBA,LRECL=133),DSNTYPE=LIBRARY
/*
/* Recompile MYPGMA
/*
//CBLCOMP EXEC IGYWC,PARM.COBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN DD DISP=SHR,DSN=MY.COBOL.Source(MYPGMA)
//COBOL.SYSPRINT DD DISP=SHR,DSN=&SYSUID..COBLIST(MYPGMA)
/*
/* Recompile MYPGMB
/*
//CBLCOMP EXEC IGYWC,PARM.COBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN DD DISP=SHR,DSN=MY.COBOL.Source(MYPGMB)
//COBOL.SYSPRINT DD DISP=SHR,DSN=&SYSUID..COBLIST(MYPGMB)
/*
/* Reanalyze SYSMDUMP data set
/*
//RUNDA   EXEC PGM=IDIDA,PARM=('DumpDSN(MY.DUMPDS)')
//SYSPRINT DD SYSOUT=*
//IDIHIST  DD DISP=SHR,DSN=MY.HISTORY.FILE
//IDILCOB  DD DISP=SHR,DSN=&SYSUID..COBLIST
/*
/* Delete temporary compiler listings PDSE
/*
//DELETE  EXEC PGM=IEFBR14
//DD1    DD DISP=(OLD,DELETE),DSN=&SYSUID..COBLIST
```

---

*Figure 38. Sample batch reanalysis job*

The above job includes the recompilation of two COBOL programs, MYPGMA and MYPGMB. These are assumed involved in the fault being reanalyzed, but their compiler listings might not have been available to Fault Analyzer during real-time analysis. Providing them to the fault reanalysis enables identification of the line of source code where the error occurred.

Note that only one program is compiled in each job step. This limitation is to facilitate the naming of the compiler listing data set member in accordance with the rules outlined in “Naming compiler listings or side files” on page 289.

## **Creating your own batch reanalysis job**

You can optionally add an IDIOPTS DD statement to your JCL. This statement supplies the name of a sequential file containing Fault Analyzer options, providing job step overrides of product and installation defaults.

Other DD statements that you can add into your JCL are described in “Pointing to listings with JCL DD statements” on page 18.

Any options that are specified in the JCL EXEC statement PARM field override options set via the IDIOPTS file.

For more information, see “User options file IDIOPTS” on page 454.

---

## Chapter 5. Performing interactive reanalysis

Interactive reanalysis provides several advantages over batch reanalysis:

- The sections of the report that are of interest can be selected and examined separately.
- Any storage area that is included in the associated minidump or SYSMDUMP can be displayed, regardless of whether it is included in the Fault Analyzer report.
- Source code information (if provided via compiler listing or side file) can be viewed in its entirety.
- This method is the only way to analyze CICS system abends.

**Note:** Whereas the information in this chapter assumes that the interactive reanalysis is performed under ISPF, it is possible to instead perform this under CICS. When doing so, restrictions might apply. These restrictions are described in “Performing interactive reanalysis under CICS” on page 206.

---

### Interactive reanalysis options

Many of the general options specified for an installation are also applicable to interactive reanalysis. See Chapter 33, “Options,” on page 451 for information about all available options and the different ways in which they can be specified.

To specify interactive reanalysis options that apply to your interactive reanalysis sessions only, select **Interactive Reanalysis Options** from the **Fault Entry List display Options** menu (for information about selecting this option in general, see “Action-bar pull-down menus” on page 61). This option opens the Interactive Reanalysis Options display as shown in Figure 39 on page 104.

## Interactive reanalysis options

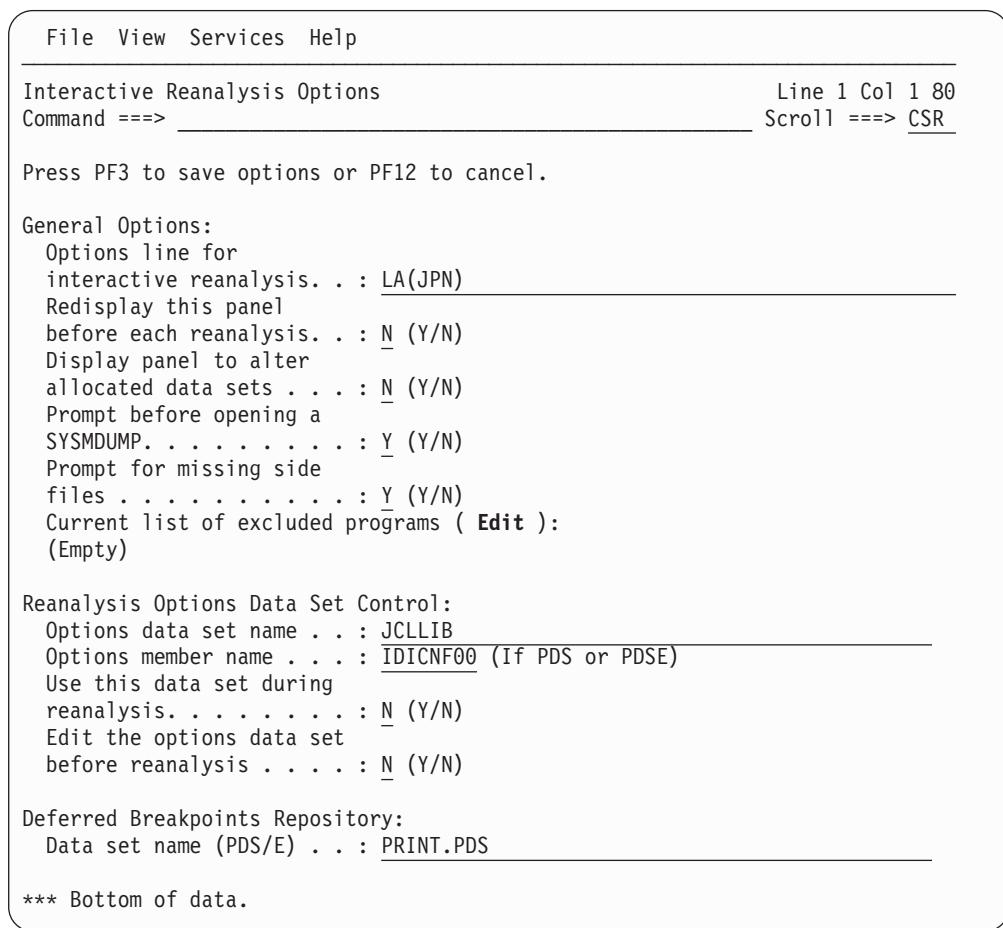


Figure 39. Sample Interactive Reanalysis Options display

The following possibilities can be specified by using this display:

### Options line for interactive reanalysis

Options that apply to all interactive reanalysis sessions that you initiate can be specified here. These options take precedence over any options that are specified through an options file (see Options data set name). The options are the equivalent of the **PARM** field options that are used by batch reanalysis jobs.

The option LA(JPN) is shown as an example on the options line in Figure 39.

### Redisplay this panel before each reanalysis

If this option is set to Y, then the Interactive Reanalysis Options display is shown each time that an interactive reanalysis is requested.

Make your changes. Then press PF3 (to continue with the current options), or press PF12 (to undo any changes). What happens next depends on the **Display panel to alter allocated data sets** option.

If this option is set to N, then the Interactive Reanalysis Options display is not shown.

### Display panel to alter allocated data sets

If this option is set to Y, then you are presented with an ISPF EDIT display screen of the pseudo JCL stream that is generated by Fault Analyzer.

```

File Edit Confirm Menu Utilities Compilers Test Help
EDIT IBMUSER.SPFTEMP1.CNTL Columns 00001 00072
Command ==> PAGE
***** **** Top of Data ****
000001 /* Data sets in this file will be allocated by Fault Analyzer.
000002 /**
000003 /* The format of each line must adhere to normal JCL syntax rules
000004 /* for DD statement specification (or comments using /* in column 1),
000005 /* with the following additional limitations:
000006 /* - Each DD statement is limited to one line
000007 /* - Each DD statement must contain the DSN= parameter - any other
000008 /* parameters will be ignored
000009 /* - Only valid data set names may appear in the DSN= parameter
000010 /* (for example, DSN=*.ddname is not permitted)
000011 /**
000012 /* Example:
000013 /* //IDILCOB DD DISP=SHR,DSN=MY.COBOL.LISTING.DATA.SET
000014 /* //           DD DISP=SHR,DSN=COMMON.COBOL.LISTING.DATA.SET
000015 /**
000016 //IDIADATA DD DISP=SHR,DSN=DA.SYSADATA
000017 //IDILCOB DD DISP=SHR,DSN=DA.LISTING.COBOL
F1=Help      F2=Split     F3=Exit      F4=Return     F5=Rfind      F6=Rchange
F7=Up        F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel

```

Figure 40. Sample interactive reanalysis pseudo JCL stream EDIT

In accordance with the instructions displayed, make your changes. Then enter the EXIT (PF3) or CANCEL (PF12) command as appropriate to initiate the interactive reanalysis.

For more information about this option, see “Data sets used for interactive reanalysis” on page 171.

If this option is set to N, then the interactive reanalysis commences without first displaying the pseudo JCL EDIT screen.

#### Prompt before opening a SYSMDUMP

Sometimes access is required to a storage location that is not contained in the saved minidump:

- During the interactive reanalysis.
- As a result of displaying storage locations from within the interactive report.

When this situation happens, if the field is set to Y, a display is shown before an associated SYSMDUMP or SVC dump data set is opened, to look for the missing storage.

An example of this display is shown in Figure 41 on page 106.

## Interactive reanalysis options

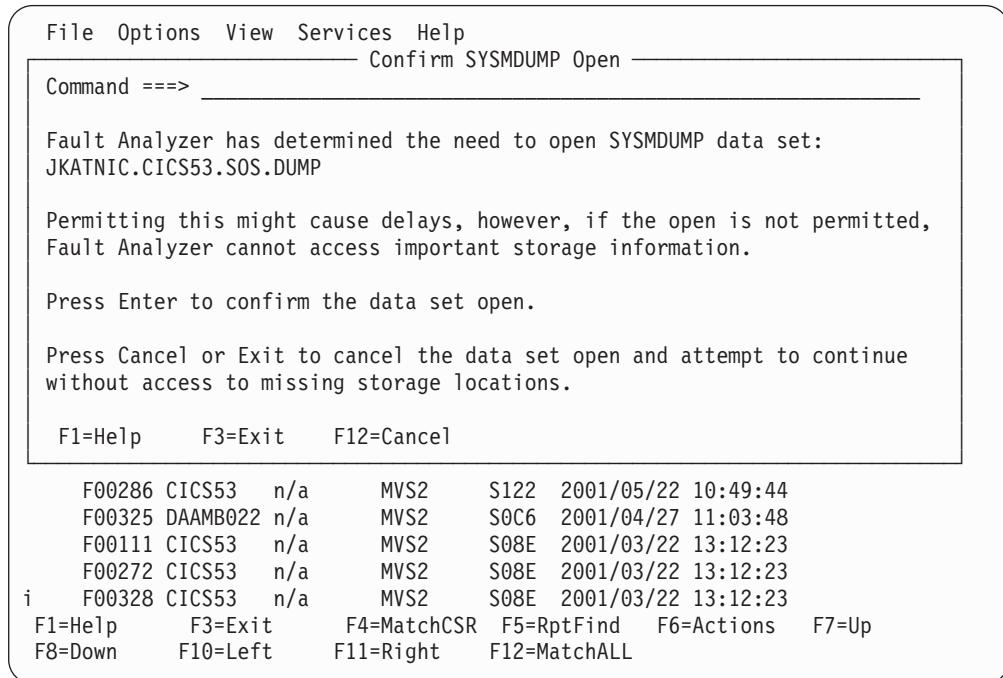


Figure 41. Sample Confirm SYSMDUMP Open display

You are only prompted at most one time during any interactive reanalysis session. If the open is canceled by entering CANCEL or EXIT, no further attempts are made to open the SYSMDUMP data set. Likewise, if the open is confirmed, Fault Analyzer checks the SYSMDUMP for all references to storage locations not contained in the minidump.

If this field is set to N, then the associated dump data set is opened if required without prompting you first.

### Prompt for missing side files

A single character (Y or N) that controls whether the Compiler Listing Not Found display is shown when no compiler listing or side file is found for a program.

The default setting of this field is based on whether any DataSets option specification exists in the IDICNFxx parmlib member for any DDnames from the following list:

IDIADATA  
IDILANGX  
IDILC  
IDILCOB  
IDILCOBO  
IDILPLI  
IDILPLIE  
IDISYSDB

If no data sets for any of these DDnames were specified in the IDICNFxx parmlib member, then the default setting of this field is 'N'. Otherwise, it is 'Y'.

By clearing this field, and pressing Enter, the value is reinitialized to its default setting.

### Current list of excluded programs (Edit)

If the "Prompt for missing side files" option is set to "Y", then the current

list of excluded program names, for which compiler listing or side file searching is not performed, is provided. The list can be modified by placing the cursor on the **Edit** point-and-shoot field and pressing Enter. The program names must be separated by one or more blanks, and can include the wildcard characters '\*' (zero, one or more characters) and '%' (a single required character). The specified program names are not case-sensitive.

The following are examples of valid program name specifications:

```
*XMAI*
PAYROLL0
SELOPT%
SUBRTN*
TZ%%C*
```

The program name exclude list can also be edited from the Compiler Listing Not Found display. For details, see “Prompting for compiler listing or side file” on page 166.

### Options data set name

This field can optionally specify the name of a PDS(E) data set in which a member (see “Options member name”) contains Fault Analyzer options. The data set and member name are used as the IDIOPTS user options file. This data set can, for example, be used if more options than fit on the options line at the top of this display are required.

#### Note:

1. The options data set is only used if the “Use this data set during reanalysis” option is set to Y.
2. Options that are specified on the options line take precedence over options that are specified in this data set.

### Options member name

This field is the member name of the data set specified in “Options data set name”.

### Use this data set during reanalysis

If this option is set to Y, then the data set and member name that are previously specified are used by Fault Analyzer during the interactive reanalysis. If it is set to N, then the data set and member name are not used.

### Edit the options data set before reanalysis

If this field is set to Y, then an ISPF EDIT display screen of the member in the options data set specified above is presented prior to commencing the interactive reanalysis. An example is shown in Figure 42 on page 108.

## Interactive reanalysis options

The screenshot shows an ISPF EDIT session window. The menu bar includes File, Edit, Confirm, Menu, Utilities, Compilers, Test, and Help. The title bar displays 'EDIT USER.JCLLIB(IDIOPTS) - 01.02'. The command line shows 'Command ==>'. The main area contains the following text:

```
***** **** Top of Data ****
000001 datasets(idiexec(user.exec))
000002 exits(listing(rexx(1x)))
***** **** Bottom of Data ****
```

At the bottom, there are function key definitions:

F1=Help	F2=Split	F3=Exit	F4=Return	F5=Rfind	F6=Rchange
F7=Up	F8=Down	F9=Swap	F10=Left	F11=Right	F12=Cancel

Figure 42. Sample options file EDIT for interactive reanalysis

Change the options data set (if required). Then enter the EXIT command (usually mapped to PF3).

### Data set name (PDS/E)

Specifies the name of the data set used for Debug Tool deferred breakpoints set via COBOL Explorer.

Standard ISPF data set name specification rules apply, that is, if the data set name is not enclosed within single quotes, then it is prefixed by the current TSO prefix.

The data set name specified need not exist, as it is allocated automatically when required. If an existing data set name is specified, then it must be the name of a PDS data set allocated RECFM=VB and LRECL=255.

A member name must not be specified.

## Initiating interactive reanalysis

To initiate interactive reanalysis, enter I against the fault history entry.

### Status pop-up display

During the interactive reanalysis that occur prior to the interactive reanalysis report being displayed, a status pop-up display, as the example below, might be presented.

The screenshot shows a terminal window titled "IBM Fault Analyzer - Fault Entry List". The command "Command ==>" is at the top. To the right, "Line 1 Col 1 80" and "Scroll ==> CSR" are displayed. Below the command, the message "Fault History File or View : 'SWILKEN.HIST'" is shown. A note about line commands follows. A table of fault entries is then presented:

Fault_ID	Dups	Job/Tran	User_ID	Sys/Job	Abend	Date	Time	MD_Pag
i SW00341		RSHPAR2	B12128	SY2	S06F	2005/07/12	14:34:06	
SW00154		PG70688P	PG7068	DEV	n/a	2005/07/07	13:00:56	
SW00137		SYSLOG	+MASTER+	SYSN	n/a	2005/07/07	09:36:13	

Below the table, the "Interactive Reanalysis Status" section is highlighted with a box. It contains the following information:

- S Current Function : le\_heap
- S Total Elapsed Time : 11 Seconds
- S Total CPU Time : 8.201 Seconds
- S

At the bottom, function keys are listed: F1=H, F2=Split, F3=Exit, F9=Swap, F12=Cancel.

*Figure 43. Sample Interactive Reanalysis Status display*

This display is presented the first time if more than 10 seconds have elapsed since the start of the interactive reanalysis, and is updated for each subsequent 10 seconds elapsed. Whereas the current function identified in the display represents a process that is internal to Fault Analyzer, it still serves as a possible indicator of loop conditions if the function identifier continually remains unchanged. Also, if the system on which the reanalysis is performed is short on CPU resources due to heavy load, the elapsed versus CPU time displayed might help to explain why the analysis appears to be performing slower than usual.

The Interactive Reanalysis Status display does not permit any user interaction and is removed automatically when the analysis is complete.

## General information about the interactive report

The interactive reanalysis report looks similar to the real-time fault analysis report, but has more functions that let you look further into the cause of the problem.

Figure 44 on page 110 shows an example of the first interactive report display from where all other parts of the interactive report can be selected:

## General information about the interactive report

The screenshot shows a window titled "Interactive Reanalysis Report". The menu bar includes "File", "View", "Services", and "Help". The main area displays the following text:

Interactive Reanalysis Report  
Command ==> Line 1 Col 1 80  
JOBNAME: IDIVPCOB SYSTEM ABEND: 0C7 MVS2 2001/12/21 13:02:25  
Scroll ==> CSR

Fault Summary:  
Module IDISCB1, program IDISCB1, source line # 31 : Abend S0C7 (Data Exception).

Select one of the following options to access further fault information:

1. Synopsis
2. Event Summary
3. Open Files
4. Storage Areas
5. Messages
6. Language Environment Heap Analysis
7. Abend Job Information
8. User Notes
9. Fault Analyzer Options

{Fault Analyzer maximum storage allocated: 1.47 megabytes.}

\*\*\* Bottom of data.

Figure 44. Sample Interactive Reanalysis Report display

A fault summary is provided at the top of the initial display, which is equivalent to the summary provided in message IDI0002I that is issued during the real-time analysis of any fault.

The individual options that can be selected from the initial display are explained in the sections that follow. The options might change between analyses of different faults. Options that are available can be entered on the command line, or the cursor can be placed on the option number and the Enter key pressed.

If the option to show help text is selected (see “Adding or removing help text” on page 76), then information about the maximum amount of allocated storage that Fault Analyzer used during the analysis is included at the bottom of the display. This amount of storage is for explicit allocations only and does not include storage for loaded modules, and so on.

The interactive report is formatted differently depending on the logical screen size used. All examples in this book are based on a screen size which is 24 lines by 80 columns, however, if your screen is larger, Fault Analyzer formats the report accordingly. This situation is true also if the screen size is dynamically resized—just press the Enter key and the report section viewed is reformatted to match the screen size.

Anywhere in the interactive report, you can use the UP (PF7), DOWN (PF8), LEFT (PF10), or RIGHT (PF11) commands to see the entire report section that is currently selected. (Note that PF10 and PF11 in the Dump Storage display are instead mapped to the PREV and NEXT commands respectively, as this display does not necessitate horizontal scrolling.)

Throughout the interactive report are tabbable fields in yellow. These are point-and-shoot fields which respond to placing the cursor on them and pressing the Enter key. What is displayed next depends on the type of information selected.

## General information about the interactive report

Some take you to other parts of the report while others display detailed information about the item selected. For example:

### Source code line or statement number

Displays the source code for the entire program as obtained from the compiler listing or side file with the selected line or statement number highlighted. In addition, disassembly of machine instructions is provided. For more information and an example of this display, refer to "Displaying source code" on page 145.

### Storage address

Displays the storage at this location in both hexadecimal and translated EBCDIC. For more information and an example of this display, refer to "Displaying storage locations" on page 147.

Although the point-and-shoot fields are defined using the ISPF color attribute YELLOW, they might actually be displayed with a different color depending on user settings. However, in this book, they are referred to as yellow fields.

Apart from storage addresses, all point-and-shoot fields can also be entered on the command line of the display. This point-and-shoot ability is especially convenient when selecting an item from a list of options.

## Exit from the interactive report

Upon exit from the interactive report using the EXIT (PF3) or CANCEL command from the Interactive Reanalysis Report display, you might be presented with a confirmation prompt as the example shown in Figure 45.

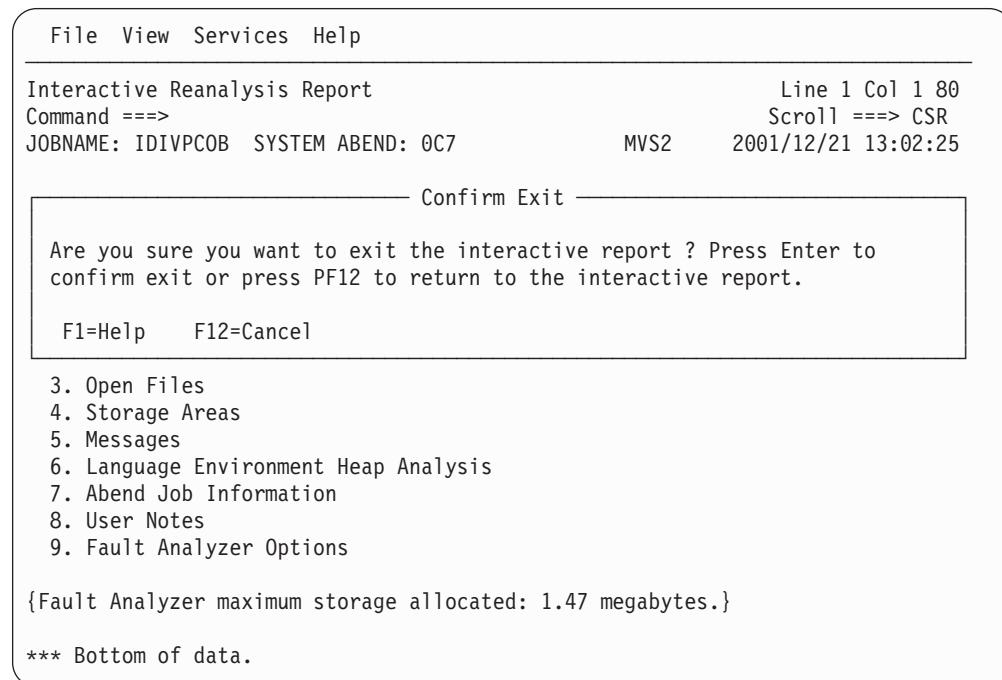


Figure 45. Sample Confirm Exit display

This display is shown to prevent unintended exit from the interactive reanalysis report by, for example, pressing PF3 once too many. To confirm the exit and return to the Fault Entry List display, press the Enter key. To instead abort the exit and resume the interactive reanalysis report, press PF12.

## Exit from the interactive report

The exit prompt panel is only displayed if the interactive reanalysis elapsed time exceeds, or is equal to, the number of seconds in effect for the InteractiveExitPromptSeconds option. For details of this option, see "InteractiveExitPromptSeconds" on page 479.

An extra prompt might be displayed if user information has been modified. For details, see "Refresh processing" on page 171.

You can exit the interactive report at any time using the ISPF jump command (for example, type =X on the command line and press Enter).

## Primary option: Synopsis

Selecting the "Synopsis" option from the initial interactive report display, results in the display of the "Synopsis" section of the report, as the example shown in Figure 46.

The screenshot shows the ISPF interface with a menu bar (File, View, Services, Help) and a title bar (Synopsis). The main area displays a system abend report. At the top, it shows the command (Command ==>), job name (JOBNAME: COBUNI), system abend code (SYSTEM ABEND: 0C9), FAE1, and timestamp (2009/07/21 14:36:43). It also shows scroll information (Line 1 Col 1 80, Scroll ==> CSR). Below this, it states: "A system abend **0C9** occurred in module COBUNI program COBUNI at offset X'5C8'. A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated with this abend and indicates that: The divisor was zero in a signed binary division. The cause of the failure was program COBUNI in module COBUNI. The COBOL source code that immediately preceded the failure was: Source Line # **000036** IF UTF-16 NOT EQUAL ' ' AND V1 / V2 > 0. The COBOL source code for data fields involved in the failure: Source Line # **000018** 01 UTF-16 PIC N(30) USAGE NATIONAL. **000020** 01 V1 PIC 9(9) BINARY VALUE 357. **000021** 01 V2 PIC 9(9) BINARY VALUE 0. Data field values at time of abend: **UTF-16** = UTF-16 UNICODE DATA **V1** = 357 **V2** = 0 \*\*\* Cause of error \*\*\*. At the bottom, it says: \*\*\* Bottom of data.

Figure 46. Sample Synopsis display

## Primary option: Event Summary

Selecting the "Event Summary" option from the initial interactive report display, results in the display of the "Event Summary" section of the report, as the example shown in Figure 47 on page 113.

## Primary option: Event Summary

File View Services Help						
Event Summary Command ==> TRANID: FRED CICS ABEND: AEIL					Line 1 Col 1 80 Scroll ==> CSR 2002/05/24 13:49:18	
{The following events are presented in chronological order.}						
Event #	Fail Type	Module Point	Program Name	EP Name	Event Location (*)	Loaded
1	Call	DFHAPLI	DFHAPLI1	n/a	P+27C4	CICS.C
2	Call	CEEPLPKA	n/a	CEECRINI	E+8B0	Not de
3	Call	CEEPLPKA	n/a	CEECRINV	E+42E	Not de
4	Call	CEEEV005	IGZCEV5	IGZCEV5	E+672	CEE.SC
5	EXEC CICS	*****	CICFRED	CICFRED	L#69 E+436	DA.TES
6	Abend AEIL	DFHAIP	DFHEIP	n/a	P+1A92	CICS.C
(*) One or more of the following abbreviations might appear in the "Event Location" column:						
F#n Source file number (refer to detailed event information for file identification)						
L#n Source file line number						
S#n Listing file statement number (refer to detailed event information for file identification)						
M+x Offset from start of load module						
P+x Offset from start of program						
E+x Offset from start of entry point						
*** Bottom of data.						
F1=Help	F3=Exit	F4=Dsect	F5=RptFind	F6=Actions	F7=Up	
F8=Down	F10=Left	F11=Right				

Figure 47. Sample Event Summary display

Individual events can be selected from this summary by placing the cursor on the event number and pressing the Enter key. The type of detailed event display that is presented if doing so is similar to the one shown in “Detailed Event Information” on page 114.

Point-and-shoot fields are provided for most of the information in the Event Location column:

- If selecting offset-type information (M+x, P+x, or E+x), the Dump Storage display is presented for the corresponding address.

For more information about the Dump Storage display, see “Displaying storage locations” on page 147.

- If selecting source or listing information (L#n or S#n), the Compiler Listing display is presented for the appropriate line or statement.

For more information about the Compiler Listing display, see “Displaying source code” on page 145.

This screen responds to the standard UP, DOWN, LEFT, and RIGHT commands, which by default are assigned to the PF7, PF8, PF10, and PF11 function keys respectively. These can be used to scroll the display horizontally or vertically as needed to see all of the information available.

Optional help text is displayed only when the top-most line of the display is shown. If the display is scrolled down any number of lines, this help text

## **Primary option: Event Summary**

disappears, but reappears if the display is scrolled to the top. For general information about help text, see “Adding or removing help text” on page 76.

The column headings are never scrolled out of view. However, if scrolling horizontally, the column headings scroll with the data below them.

## **Detailed Event Information**

An Event Details display is presented when an event is selected from the Event Summary display. An example is shown in Figure 48 on page 115.

## Primary option: Event Summary

```

File View Services Help
Event 1 of 1: Abend S0C7 *** Point of Failure *** Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBPERF6 SYSTEM ABEND: 0CF FAE1 2009/07/21 16

Abend Code. . . . . : S0CF
Program-Interruption Code . : 000F (HFP-Divide Exception)
The divisor in an HFP division had a zero fraction.

The source code below was executed via the following sequence of PERFORM
statements: 1
Source
Line #
000041      PERFORM CALC THRU CALC-EXIT
000069      PERFORM REDO THRU REDO-EXIT.
000082      PERFORM UNDO THRU UNDO-EXIT
000091      PERFORM ABEND.

COBOL Source Code:
Source
Line #
000097      ABEND.
000098      COMPUTE FIELD-4 ROUNDED =
000099          ELEMENT-4(3) / ELEMENT-2(5)

Data Field Declarations:
Source
Line #
000016          05 ELEMENT-2 COMP-2.
000018          05 ELEMENT-4 PIC 999999 COMP-4.
000030          01 FIELD-4 PIC 999999.

Data Field Values:
ELEMENT-2(5) = 0.000000e+00 *** Cause of error ***
ELEMENT-4(3) = 222
FIELD-4      = X'000000000000'

The listing file used for the above was found in
JERRYBL.LISTING.COBOL(COBPERF6).

Load Module Name. . . . . : SYS09202.T161638.RA000.COBPERF6.GOSET.H01(COBPER
At Address. . . . . : 16B00988
Load Module Length. . . . : X'1678'
Link-Edit Date and Time . : 2009/07/21 16:16:40

Program and Entry Point Name: COBPERF6
At Address. . . . . : 16B00988 (Module COBPERF6 offset X'0')
Program Length. . . . . : X'A4A'
Program Language. . . . . : COBOL (Compiled using IBM Enterprise COBOL for
z/OS and OS/390 V4 R1 M0 on 2009/07/21 at
16:16:39)
Compiler Options Used . . . : NOADATA ADV APOST ARITH(COMPAT) NOAWO
BUFSIZE(4096) NOCICS CODEPAGE(1140) NOCOMPILE($)
NOCURRENCY DATA(31) NODATEPROC DBCS NODECK
NODIAGTRUNC NODLL NODUMP DYNAM NOEXIT
NOEXPORTALL NOFASTSRT FLAG(I,I) NOFLAGSTD
INTDATE(ANSI) LANGUAGE(EN) LIB LINECOUNT(60)
LIST MAP NOMDECK NONAME NSYMBOL(NATIONAL)
NONUMBER NUMPROC(NOPFD) OBJECT NOOFFSET

```

Figure 48. Sample Event Details display for point of failure (Part 1 of 2)

## **Primary option: Event Summary**

NOOPTIMIZE OUTDD(SYSOUT) PGMNAME(COMPAT) RENT  
 RMODE(AUTO) SEQUENCE SIZE(MAX) SOURCE SPACE(1)  
 NOSQL SQLCCSID SSRANGE NOTERM NOTEST NOThread  
 TRUNC(STD) NOVBREF NOWORD XMLPARSE(XMLSS)  
 XREF(FULL) YEARWINDOW(1900) ZWB

**Binary Optimizer. . . . .** : Automatic Binary Optimizer for z/OS V1 R0 M0  
 optimized COBPERF6 on 2015/09/03 at 12:57:14  
 using ARCH(10)

**Machine Instruction . . . . .** : 6D008094 DD FR0,148(,R8)  
**At Address. . . . .** : **16B010D8** (Program COBPERF6 offset X'750')

**AMODE . . . . .** : 31

**Failing Operand . . . . .** : Second operand

**First Operand (FR0) . . . . .** : 42DE0000 00000000

**Second Operand Address. . . . .** : **16BB4164** (Module COBPERF6 program COBPERF6  
 WORKING-STORAGE SECTION BLW=0000 + X'94', symbol  
 ELEMENT-2, source line # 16 - 433820 bytes of  
 storage addressable)

**Second Operand Length . . . . .** : 8

**Second Operand Storage. . . . .** : 00000000 00000000 \*.....\*

**Program Status Word (PSW) . . . . .** : 078D2000 **96B010DC**

**General Purpose Registers (AMODE: 64 31 24 , Bytes: Dec Hex ):** **3**  
**R0: 16B96190** (Module COBPERF6 program COBPERF6 LOCAL-STORAGE SECTION  
 BLK=0001 + X'F80')

**R1: 16B00C6E** (Module COBPERF6 program COBPERF6 + X'2E6')

**R2: 800000DE** (1826 bytes of storage addressable)

**R3: 00000000** (2048 bytes of storage addressable)

**R4: 16B00E66** (Module COBPERF6 program COBPERF6 + X'4DE')

**R5: 40000000** (Storage invalid)

**R6: 00000000** (2048 bytes of storage addressable)

**R7: 00000000** (2048 bytes of storage addressable)

**R8: 16BB40D0** (Module COBPERF6 program COBPERF6 WORKING-STORAGE SECTION  
 BLW=0000 + X'0', symbol FILLER, source line # 10 )

**R9: 16B90448** (580536 bytes of storage addressable)

**R10: 16B00ABC** (Module COBPERF6 program COBPERF6 + X'134')

**R11: 16B00C98** (Module COBPERF6 program COBPERF6 + X'310')

**R12: 16B00A84** (Module COBPERF6 program COBPERF6 + X'FC')

**R13: 16B94030** (565200 bytes of storage addressable)

**R14: 96B00ECC** (Module COBPERF6 program COBPERF6 + X'544', source line # 39 )

**R15: 96B577F0** (Module IGZCPAC + X'40C48')

**Floating-Point Registers:**  
**R0: 42DE0000 00000000**    **R2: 00000000 00000000**  
**R4: 00000000 00000000**    **R6: 00000000 00000000**

**Associated Messages**

**CEE3215S** The system detected a floating-point divide exception (System Completion Code=0CF).

**Associated Storage Areas**

\*\*\* Bottom of data.

Figure 49. Sample Event Details display for point of failure (Part 2 of 2)

### Notes:

- 1** The COBOL PERFORM traceback might only appear if the code has not been inlined by the compiler. This is the case for optimized programs and non-optimized programs which have only one PERFORM of a paragraph or section.

- 2 Automatic Binary Optimizer for z/OS (ABO) information will only appear here if a COBOL program has been optimized by ABO.
- 3 The general purpose registers are initially displayed in accordance with the event AMODE, or defaults to AMODE 31 if no event AMODE has been determined. However, by selecting the AMODE 64, 31 or 24 point-and-shoot fields, the register display changes accordingly. In addition, the default display of the number of bytes in the register description is decimal, but this can be changed to hexadecimal or back to decimal again by selecting the Dec or Hex point-and-shoot fields.

All information that is associated with the currently selected event is either already included in the displayed information, or a point-and-shoot link to the information is provided in yellow. Such links include message and abend codes for which an explanation can be provided if selected (refer to “Expanding messages and abend codes” on page 144) and associated storage areas (refer to “Displaying associated storage areas” on page 140).

To select the previous or next event from the one currently displayed (when the fault includes more than one event), point-and-shoot links in yellow are provided at the bottom of the display. Simply place the cursor on one of these and press the Enter key.

## Primary option: Open Files

Selecting the “Open Files” option provides you with a display that lists all open files which could not be associated with any particular event, as well as files that might be listed in the detail section of the report for individual events. An example of the open file list is shown in Figure 50.

```
File View Services Help
System-Wide Open Files Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBTSTC SYSTEM ABEND: 0C9 FAE1 2007/09/18 09:02:09

Event 1 Program COBTSTC Open Files
File Name . . . . . : OUTDD

Non-Event-Related Open Files
File Name . . . . . : SYSOUT

*** Bottom of data.
```

Figure 50. Sample System-Wide Open Files display

The listed file names are point-and-shoot fields that you can place the cursor on and press the Enter key in order to see the associated detailed file information. For example, if selecting the file OUTDD from the above sample display, you might see the “File Information” display shown in Figure 51 on page 118.

## Primary option: Open Files

(You return from the Open Files display by entering the Exit (PF3) command.)

```

File View Services Help
File Information Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBTSTC SYSTEM ABEND: 0C9 FAE1 2007/09/18 09:02:09

File Name . . . . . : OUTDD
Data Set Name . . . . . : SWILKEN.OUT80S 1
File Attributes . . . . . : ORGANIZATION=SEQUENTIAL, ACCESS MODE=SEQUENTIAL,
                           RECFM=VARIABLE BLOCKED SPANNED
Last I/O Function . . . . : WRITE
Open Status . . . . . : OUTPUT
File Status Code. . . . . : 0

Last Record Written -2. . : RDW=001C0100 Segment data length 24, variable re
Address Offset Hex EBCDIC / ASCII
16599FC0 C9C9C9F9 F9F94040 40404040 40404040 *III999 *
16599FD0 +10 40404040 40404040 * * *

Last Record Written -1. . : RDW=00240300 Segment data length 32, variable re
Address Offset Hex EBCDIC / ASCII
16599FE0 40404040 40404040 40404040 40404040 * * *
Line 16599FF0 same as above

Last Record Written . . . : RDW=001C0200 Segment data length 24, variable re
Address Offset Hex EBCDIC / ASCII
16599EF0 40404040 40404040 40404040 40404040 * * *
16599F00 +10 40404040 40404040 * * *

Current Record Build Area : RDW is zero, no length assigned yet
Address Offset Hex EBCDIC / ASCII
1659AFA4 D1D1D1C1 C1C14040 40404040 40404040 *JJJAAA *
1659AFB4 +10 40404040 40404040 40404040 40404040 * *
Lines 1659AFC4-1659AFD4 same as above
1659AFE4 +40 40404040 40404040 40404040 40406E6E * >>*
1659AFF4 +50 40404040 * * *

Associated File Control Blocks 2

*** Bottom of data.

```

Figure 51. Sample File Information display

If the data set shown at **1** is QSAM, VSAM, or IAM, and exists, then it is presented as a point-and-shoot field, that if selected, shows a selection menu from where either Edit, View or Browse can be chosen. If IBM File Manager for z/OS is available, then it is used to perform the requested function against the data set. Otherwise, if the data set is QSAM non-spanned record format, then ISPF is invoked.

For more information about the presentation of open file buffers, see "Open file record information" on page 197.

An "Associated File Control Blocks" point-and-shoot field might be available at the bottom of the File Information display, as shown at **2** above. By placing the cursor on this field and pressing Enter, the Associated File Control Blocks display is presented. An example of this display is shown in Figure 52 on page 119.

## Primary option: CICS Information

```

File View Services Help
Associated File Control Blocks Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBTSTC SYSTEM ABEND: OC9 FAE1 2007/09/18 09:02:09

Data Extent Block (DEB) at Address 008BCD84 :
Address Offset Hex EBCDIC / ASCII
008BCD84 038C2CF0 10000000 E8000000 * ...0....Y...
008BCD90 +C 0F001100 01000000 FF000000 8F01A038 *.....*.
008BCDA0 +1C 048BCD60 10F41AE0 000000FF 000000FF *...-.4.\....*
008BCDB0 +2C 000E000F 00010001 00000000 00000000 *.....*.
008BCDC0 +3C 00000054 F3C2C1D9 C2D70000 00000000 *....3BARBP....*
008BCDD0 +4C 00000000 00000000 00000000 00000000 *.....*.
008BCDE0 +5C 00000000 00000000 E2E6C1D4 00000000 *.....SWAM....*
008BCDF0 +6C 00000218 008BCE08 500000E6 008BD6B0 *.....&..W..O.*.
008BCE00 +7C 008BD140 *..J *.

Data Control Block (DCB) at Address 0001A038 :
Address Offset Hex EBCDIC / ASCII
0001A038 1656C858 00000000 * ..H.....*
0001A040 +8 00FF0000 0EF15026 002FBEB9 07022FE8 *.....1&....o...Y*
0001A050 +18 00004000 00006B70 E6000001 5801E8B4 *.. .,,.W....Y.*.
0001A060 +28 007C0048 008BCD84 92C9D870 00C8B348 *..@....dkIQ..H...*

```

Figure 52. Sample Associated File Control Blocks display

## Primary option: CICS Information

Selecting the “CICS Information” option provides you with a display of information that is related to CICS, as the example shown in Figure 53.

```

File View Services Help
CICS Information Line 1 Col 1 80
Command ==> Scroll ==> CSR
TRANID: CS31 CICS ABEND: ASRA FAE2 2005/08/16 12:06:24

CICS Release. . . . . : 0620
Application ID. . . . . : QXPE262X
CICS Transaction ID . . . . : CS31
CICS Task Number. . . . . : 00027
CICS Terminal ID. . . . . : SAMA
CICS Terminal Netname . . . . : n/a

Select one of the following:
1. CICS Control Blocks
2. CICS Transaction Storage Summary
3. CICS Transaction Storage
4. Last CICS 3270 Screen Buffer
5. Last CICS 3270 Screen Buffer Hex
6. Summarized CICS Trace
7. CICS Trace Formatting
8. CICS Recovery Manager
9. CICS Levels, Commareas, and Channels

*** Bottom of data.

```

Figure 53. Sample CICS Information display

The CICS Information display provides options, that can be entered on the command line or selected using the cursor, to extra CICS information, such as the following.

## Primary option: CICS Information

### CICS Transaction Storage Summary

Selecting the "CICS Transaction Storage Summary" link provides you with a display which provides information about the storage areas owned by the current CICS transaction, as the example shown in Figure 54.

CICS Transaction Storage Summary - All Tasks						Line 1 Col 1 80
Command ==> Current All - TRANID: CFA CICS ABEND: FLT1						Scroll ==> CSR
Address	Length	Type	ID	Task #	Possible Overlay	FAE1 2013/07
00100008	00002D60		NEWC	0002915C	n/a	
00103008	00000650		CECI	0002916C	n/a	
00104000	000029F0	USER24			None detected	
	Total:	000029F0	USER24			
19320008	00000030		NEWC	0002915C	n/a	
193201E8	000000C0		NEWC	0002915C	n/a	
193202B8	00003AD0		NEWC	0002915C	n/a	
19323D98	0000A770		NEWC	0002915C	n/a	
19330008	000001C0		CECI	0002916C	n/a	
193301E8	000000C0		CECI	0002916C	n/a	
193302B8	00007D00		CECI	0002916C	n/a	
19337FC8	00000120		CECI	0002916C	n/a	
193380F8	000004F0		CECI	0002916C	n/a	
1934B170	00000FD0	USER31			None detected	
19340660	000000E0	USER31			None detected	
19340910	0000A860	USER31			None detected	

Figure 54. Sample CICS Transaction Storage Summary display

The CICS Transaction Storage Summary display can be important when a transaction's storage has been overlaid by another task, as it helps identify transactions whose storage was near or adjacent to the overlay. When Fault Analyzer is invoked to analyze a CICS transaction abend, the storage allocations of all other concurrently running tasks are gathered. This information can then be displayed, during interactive reanalysis, in the CICS transaction storage summary display.

The display has two point-and-shoot fields, "Current" and "All", which toggle the display accordingly.

### Last CICS 3270 Screen Buffer

Selecting the "Last CICS 3270 Screen Buffer" link provides you with a display of the last 3270 screen buffer written by CICS as the example shown in Figure 55 on page 121.

```

File View Services Help
Last CICS 3270 Screen Buffer
Command ==>
TRANID: SK00      CICS ABEND: ASRA
Line 1 Col 1 80
Scroll ==> CSR
CICS02 2001/10/17 10:03:37

Column
Row ---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+
1 sk00*** CICB0CB0 - START OF PROGRAM.....
2 .....
3 .....
4 .....
5 .....
6 .....
7 .....
8 .....
9 .....
10 .....
11 .....
12 .....
13 .....
14 .....
F1=Help   F3=Exit   F5=RptFind   F6=Actions   F7=Up   F8=Down
F10=Left  F11=Right

```

Figure 55. Sample Last CICS 3270 Screen Buffer display

All non-printable characters are shown as periods (.).

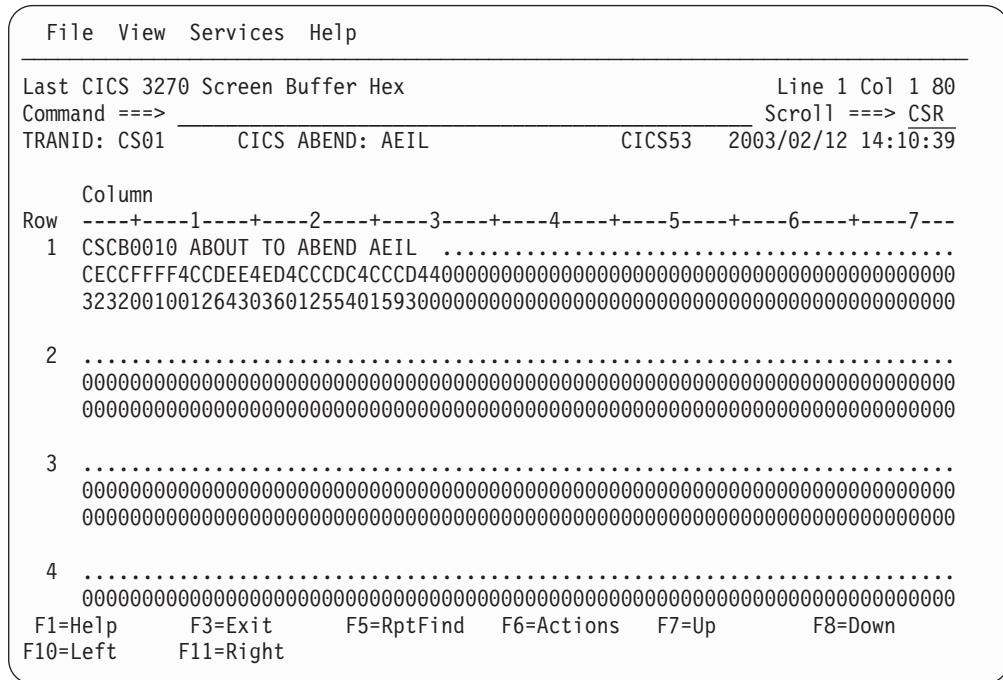
It is not always possible for Fault Analyzer to provide CICS 3270 screen buffer information. For example, if there are indications of VTAM session errors on the terminal, or if one of the following CICS abends have occurred (\* indicates a wildcard character), then no 3270 screen buffer information is provided:

- AEXZ
- AKC3
- AKCT
- AKK\*
- ATC\*
- ATN\*
- AZCT
- AZI\*
- AZTS

## Last CICS 3270 Screen Buffer Hex

Selecting the “Last CICS 3270 Screen Buffer Hex” point-and-shoot field provides you with the same display as the “Last CICS 3270 Screen Buffer” point-and-shoot field, but in format that permits viewing of the hexadecimal values of all characters, as the example shown in Figure 56 on page 122.

## **Primary option: CICS Information**



*Figure 56. Sample Last CICS 3270 Screen Buffer Hex display*

All non-printable characters are shown as periods (.).

The conditions under which information is available are the same as for “Last CICS 3270 Screen Buffer” on page 120.

## Summarized CICS Trace

Selecting the “Summarized CICS Trace” link provides you with a display of the CICS trace table as the example shown in Figure 57 on page 123.

```

File View Services Help
Summarized CICS Trace Line 1 Col 1 80
Command ==> Scroll ==> CSR
TRANID: CS31 CICS ABEND: ASRA FAE2 2005/08/16 12:06:24

00027 QR AP EA00 TMP ENTRY LOCATE PFT,DFHCICST
00027 QR XS 0701 XSRC ENTRY CHECK_CICS_RESOURCE CS31,TRANSATTACH,EXECUTE
00027 QR PG 0901 PGPG ENTRY INITIAL_LINK CSCB0310
00027 QR AP 1940 APLI ENTRY ESTABLISH_LANGUAGE CSCB0310,17F00160,97F001
00027 QR AP 1940 APLI ENTRY START_PROGRAM CSCB0310,CEDF,FULLAPI,EX
00027 QR AP 00E1 EIP ENTRY SEND-TEXT
Called-from-address 17F0051E : Module CSCB0310 program CSCB0310 + X'396', sou
00027 QR AP 00E1 EIP EXIT SEND-TEXT OK

00027 QR AP 00E1 EIP ENTRY HANDLE-ABEND
Called-from-address 17F0055C : Module CSCB0310 program CSCB0310 + X'3D4', sou
00027 QR PG 0701 PGHM EXIT SET_ABEND/OK
00027 QR AP 00E1 EIP EXIT HANDLE-ABEND OK

00027 QR AP 0790 SRP *EXC* PROGRAM_CHECK
00027 QR DU 0601 DUTM EXIT INQUIRE_SYSTEM_DUMPSPACE/EXCEPTION_DUMPSPACE_NOT
00027 QR DU 0601 DUTM EXIT LOCATE_SYSTEM_DUMPSPACE/OK FFFFFFFF,1,0,0,NO,YE
00027 QR DU 0601 DUTM EXIT COMMIT_SYSTEM_DUMPSPACE/OK

```

Figure 57. Sample Summarized CICS Trace display

The standard CICS trace table is enhanced by Fault Analyzer for greater ease of use. Information is added to indicate the call point origin addresses, including the module name, CSECT name, offset within CSECT, and source line or listing statement number if available.

As in all other sections of the interactive report, source line numbers or listing statement numbers are selected by placing the cursor on them and pressing Enter. This provides a full source listing display as shown in “Displaying source code” on page 145.

## CICS Trace Formatting

Selecting the “CICS Trace Formatting” link provides you with a display that permits specific selection parameters to be entered for the CICS trace, as the example shown in Figure 58 on page 124.

## Primary option: CICS Information

The screenshot shows a terminal window titled "CICS Trace Selection Parameters". The window contains several input fields for specifying trace selection parameters. On the right side of the window, there is a status bar showing "ine 1 Col 1 80", "roll ==> CSR", and the date and time "5/22 14:22:19". Below the window, two lines of text are displayed: "7. CICS Levels, Commareas, and Channels" and "\*\*\* Bottom of data.".

Format . . . . .	A (Abbrev/Short/Full)
Exception Only . . .	N (Yes/No)
Sequence Start . . .	000001
End . . .	000375
Highlight Interval . . . . .	(0-99.999999999 secs)
Task IDs . . . . .	00027
KE Task Numbers	
Terminal IDs . . . . .	Caps Y
Transaction IDs . . . . .	Caps Y
Time Start . . . . .	(HHMMSS)
End . . . . .	(HHMMSS)
Domain/Point IDs	
Exclude IDs	

Figure 58. Sample CICS Trace Selection Parameters display

The following parameters can be specified:

### Format

Specifies the level of trace formatting:

- A The abbreviated form of the trace, with one line per entry.
- S The short formatted display, consisting of the abbreviated entry plus fully formatted parameter list, return address, time, and interval.
- F The fully formatted display of all the data in each entry.

### Exception Only

Specifies that only exception entries in the internal trace are to be displayed.

### Sequence Start/End

Specifies which sequence numbers are to be selected (sequence numbers start at 1 and are up to 6 digits in length). You can specify Start, End or both.

### Highlight Interval

Specifies the interval between internal trace entries after which entries are highlighted (with an asterisk).

### Task IDs

Specifies up to five task identifiers whose trace entries are to be displayed.

An ID value can be one of:

- Any number up to 5 digits in length.
- Any of JAS, J01 through J99.
- III, TCP, or DST.
- A two character domain ID of the attaching domain (for non-TCA tasks).

### KE Task Numbers

Specifies up to 5 four hexadecimal digit kernel task numbers to be displayed.

**Terminal IDs**

Specifies the terminal identifiers of up to five terminals for which trace entries are to be displayed. If any terminal IDs contain lowercase, set Caps to N and enter all IDs as case-sensitive. When Caps is Y, all entries are translated to uppercase.

**Transaction IDs**

Specifies the transaction identifiers of up to five transactions for which trace entries are to be displayed. If any transaction IDs contain lowercase, set Caps to N and enter all IDs as case-sensitive. When Caps is Y, all entries are translated to uppercase.

**Time Start/End**

Specifies the time period for which trace entries are to be displayed. You can specify Start, End or both.

**Domain/Point IDs**

Specifies up to 5 two-character domain IDs, and optionally followed by a trace point ID within that domain. These IDs are included. Here are the domain IDs:

AP BA BR CC DD DE DH DM DP DS DU EJ EM EX GC IE II IS KE LC  
LD LG LM ME ML MN NQ OT PA PG PI PT RL RM RS RX RZ SH SJ SM  
SO ST TI TR TS US WB W2 XM XS

The trace point ID can be up to four hexadecimal digits in length (if fewer than 4 digits are entered, then it is treated as a generic value).

If no domain or trace point IDs are specified in this selection parameter field, then the default is to include all trace entries. The exception is any trace entries that are excluded by way of the Exclude IDs selection parameter. Otherwise, only trace records that match the specified IDs (which are not also excluded via the Exclude IDs selection parameter) are shown.

**Exclude IDs**

Specifies up to 5 two-character domain IDs (same values as for Domain/Point IDs), optionally followed by a trace point ID within that domain. These IDs are excluded. The trace point ID can be up to four hexadecimal digits in length (if fewer than 4 digits are entered, then it is treated as a generic value).

After you make any optional changes to parameters, press Enter to view the CICS trace. An example of the CICS Trace display is shown in Figure 59 on page 126.

## Primary option: CICS Information

File View Services Help				
CICS Trace Command ==> Abbrev Short Full - TRANID: CS65 IDISNAP CALL				Entry 1 Col 1 80 Scroll ==> CSR FAE2 2
00027 QR	SM 0D02 SMMF	EXIT FREEMAIN/OK	CICS storage at 174E7748	
00027 QR	SM 0C01 SMMG	ENTRY GETMAIN	2000,NO,00,CICS	
00027 QR	SM 0C02 SMMG	EXIT GETMAIN/OK	174CF028	
00027 QR	SM 0D01 SMMF	ENTRY FREEMAIN	174CD018,CICS	
00027 QR	SM 0D02 SMMF	EXIT FREEMAIN/OK	CICS storage at 174CD018	
00027 QR	SM 0C01 SMMG	ENTRY GETMAIN	2000,NO,00,CICS	
00027 QR	SM 0C02 SMMG	EXIT GETMAIN/OK	174D1038	
00027 QR	SM 0D01 SMMF	ENTRY FREEMAIN	174CF028,CICS	
00027 QR	SM 0D02 SMMF	EXIT FREEMAIN/OK	CICS storage at 174CF028	
00027 QR	SM 0C01 SMMG	ENTRY GETMAIN	2000,NO,00,CICS	
00027 QR	SM 0C02 SMMG	EXIT GETMAIN/OK	174CD018	
00027 QR	SM 0D01 SMMF	ENTRY FREEMAIN	174D1038,CICS	
00027 QR	SM 0D02 SMMF	EXIT FREEMAIN/OK	CICS storage at 174D1038	
00027 QR	SM 0C01 SMMG	ENTRY GETMAIN	2000,NO,00,CICS	
00027 QR	SM 0C02 SMMG	EXIT GETMAIN/OK	174CF028	
00027 QR	SM 0D01 SMMF	ENTRY FREEMAIN	174CD018,CICS	
00027 QR	SM 0D02 SMMF	EXIT FREEMAIN/OK	CICS storage at 174CD018	
00027 QR	SM 0C01 SMMG	ENTRY GETMAIN	2000,NO,00,CICS	
00027 QR	SM 0C02 SMMG	EXIT GETMAIN/OK	174E7748	

Figure 59. Sample CICS Trace display

Initially, the trace is formatted in accordance with the settings made on the CICS Trace Selection Parameters display. However, at any time while the trace is displayed, the format can be changed by selecting one of the point-and-shoot fields provided at the top of the display. The top-most trace entry that is displayed remains at the top, regardless of the format selected.

## CICS Levels, Commareas, and Channels

Selecting the “CICS Levels, Commareas, and Channels” point-and-shoot field provides you with a display as the example shown in Figure 60.

File View Services Help				
CICS Levels, Commareas, and Channels Command ==> TRANID: CONT CICS ABEND: CVER				Line 1 Col 1 80 Scroll ==> CSR FAE1 2014/09/17 14:39:58
Lnk	Ev	Commarea	Created	Passed f
#	#	Type Program Address Len Channel(s)	Container name	Channel
1	4	EXEC CONTEST	PASS1001	CONT1002
			LOCAL102	CONT1001
			COMMONCHANNEL	CONT1002
			LOCAL101	CONT1001
			PASS2001	COMMONCONTAINER
2	8	EXEC CONTEST2	LOCAL202	CONT1002
			CONT2001	PASS1001
			COMMONCHANNEL	CONT2002
			LOCAL201	CONT2001
			PASS3001	CONT3001-1MB
3	12	EXEC CONTEST3		PASS2001

Figure 60. Sample CICS Levels, Commareas, and Channels display

## Primary option: CICS Information

Selecting the container name point-and-shoot field provides information about the container type. For example, selecting the CONT2002 point-and-shoot field at **1**:

```

CICS Container                               Line 1 Col 1 80
Command ==> _____                           Scroll ==> CSR
TRANID: CONT      CICS ABEND: CVER          FAE1      2014/09/17 14:39:58

Container CONT2002 at address 177E33A0 has a length of X'1024'

Segment      Segment    Segment Preview   Segment Preview
Address      Offset     (EBCDIC)        (Hex)
17912028    X'000000'  CONT2002/L202...  C3D6D5E3 F2F0F0F2 61D3F2F0 F24B
1790C328    X'000FD8'  -----          60606060 60606060 60606060 6060

```

*Figure 61. Sample CICS container display*

For some standard CICS containers, a description of the container is also provided. For an example, see Figure 62.

```

Container Data                               Line 1 Col 1 80
Command ==> _____                           Scroll ==> CSR
TRANID: COBA      CICS ABEND: ATCV          FAE1      2005/04/06 12:32:44

DFHWS-BODY Contains the body section of the SOAP envelope. Typically, the
application will modify the contents.

2
Data Length X'3FE' Format XML Data

Address  Offset   Hex          ASCII / EBCDIC
16A4B788    3C534F41 502D454E *      <SOAP-EN*
16A4B790    +8 563A426F 64793E20 20202020 20202020 *V:Body> *
16A4B7A0    +18 20202020 20202020 20202020 20202020 *      *
Lines 16A4B7B0-16A4B7C0 same as above
16A4B7D0    +48 20202020 20200D0A 3C434943 53564552 *      ..<CICSVER*
16A4B7E0    +58 324F7065 72617469 6F6E3E20 20202020 *20peration> *
16A4B7F0    +68 20202020 20202020 20202020 20202020 *      *
Lines 16A4B800-16A4B810 same as above
16A4B820    +98 20202020 20202020 0D0A3C63 6F6D6D61 *      ..<comma*
16A4B830    +A8 7265613E 20202020 20202020 20202020 *rea> *
16A4B840    +B8 20202020 20202020 20202020 20202020 *      *
Lines 16A4B850-16A4B860 same as above

```

*Figure 62. Sample standard CICS container display*

The character-interpreted section on the right-hand side of the hex data is automatically displayed as either EBCDIC or ASCII.

Selecting the "Format XML Data" point-and-shoot field at **2** shows the formatted XML data:

## Primary option: Messages

```
XML Formatter Line 1 Col 1 80
Command ===> Scroll ===> CSR
TRANID: COBA CICS ABEND: ATCV FAE1 2005/04/06 12:32:44

<SOAP-ENV:Body>
  <CICSVER20peration>
    <commarea>
      <DisplayOrUpdate>
        T
      </DisplayOrUpdate>
      <LinkRC>
        0
      </LinkRC>
      <msg1>
        Message 1
      </msg1>
      <msg2>
        Message 2
      </msg2>
      <msg3>
        Message 3
      </msg3>
```

Figure 63. Sample CICS XML formatter display

## Primary option: Messages

Messages written to the system console that were not identified as belonging to any specific event are listed under the heading “Messages”. Also included are any LE messages identified for specific events. Individual messages can be selected for their explanation by placing the cursor on the message number and pressing the Enter key.

An example of the display presented when selecting the “Messages” link is shown in Figure 64 on page 129.

## Primary option: DB2 Information

```
File View Services Help
System-Wide Messages                                         Line 1 Col 1 80
Command ==>                                                 Scroll ==> CSR
JOBNAME: P00398     SYSTEM ABEND: 0CB                      MVS2      2003/03/24 13:54:05

Event 5 Messages

CEE3211S Job-specific text not available

Non-Event-Related Messages

$HASP375 P00398   ESTIMATED  LINES EXCEEDED
$HASP375 P00398   ESTIMATE EXCEEDED BY          10,000  LINES
$HASP375 P00398   ESTIMATE EXCEEDED BY          20,000  LINES
*** Bottom of data.

F1=Help      F3=Exit      F5=RptFind    F6=Actions   F7=Up       F8=Down
F10=Left     F11=Right
```

Figure 64. Sample System-Wide Messages display

## Primary option: DB2 Information

Selecting the “DB2 Information” option provides you with a display of information that is related to DB2 as the example shown in Figure 65.

```
File View Services Help
DB2 Information                                         Line 1 Col 1 80
Command ==>                                                 Scroll ==> CSR
JOBNAME: DAC2DH07  SYSTEM ABEND: 806                  MVS2      2002/11/26 09:44:50

DB2 Subsystem DBT6

DB2 Version . . . . . : V6R1M0
Plan Name . . . . . : DAC2DH (Bound 2004/04/25 12:13:14)
Plan Owner. . . . . : HARRIDA
Package Name. . . . . : DAPNDH11 (Created 2002/07/30 10:06:08, bound
                           2004/04/25 12:13:13)
Package Collection ID . . . . : DAC2DHPK
Package Owner . . . . . : HARRIDA
Package Creator . . . . . : HARRIDA
Package Version . . . . . : HV01
Package Qualifier . . . . . : DSN8610
Database Request Module Name: CTEST.DB2.DBRMLIB.DATA(DAPNDH11) (Precompiled
                           2004/04/25 12:07:57)
Consistency Token . . . . . : X'177522E41D026D08'
Primary Authorization ID.. : HARRIDA
Current SQL ID. . . . . : HARRIDA
```

Figure 65. Sample DB2 Information display (Part 1 of 3)

## Primary option: DB2 Information

```
Last Executed SQL Statement:  
List  
Stmt #  
000227 EXEC SQL FETCH HVAR1 INTO :HVTABLE  
  
Fault Analyzer Event #. . . . : 7 (Program DAPNDH11)  
Declare Cursor Stmt No. . . . : 133  
Declare Cursor Stmt . . . . . : DECLARE HVAR1 CURSOR FOR SELECT HVARKEY ,  
                                VCHAR300 , DEC9 , LINT , CHARHEX , TIMESTMP  
                                FROM HVARS  
Open Cursor Stmt No . . . . . : 166  
Open Cursor Stmt. . . . . : OPEN HVAR1  
  
Output Host Variables:  
Name and Data Type. . . . . : HVTABLE.HVARKEY CHARACTER(6)  
At Address. . . . . . . . . : 0007EA26  
Data Value. . . . . . . . . : 000003  
  
Name and Data Type. . . . . : HVTABLE.VCHAR300 VARCHAR(300)  
At Address. . . . . . . . . : 0007EA2C  
Data Value. . . . . . . . . : This is record 3  
=====  
=====101  
=====  
=====  
201  
=====  
=====>  
  
Name and Data Type. . . . . : HVTABLE.DEC9 DECIMAL(9,4)  
At Address. . . . . . . . . : 0007EB5A  
Data Value. . . . . . . . . : 12345.6789  
  
Name and Data Type. . . . . : HVTABLE.LINT INTEGER  
At Address. . . . . . . . . : 0007EB74  
Data Value. . . . . . . . . : 214748364  
  
Name and Data Type. . . . . : HVTABLE.CHARHEX CHARACTER(21)  
At Address. . . . . . . . . : 00082A8C  
Data Value. . . . . . . . . : *** Data format error: Character string contains  
                           non-printable character at offset X'B' ***  
Host Variable Storage:  
Address   Offset   Hex                                                          EBCDIC / ASCII  
00082A8C        E4959799 8995A381 82938522 83888199 *Unprintable.char*  
00082A9C        +10   8183A385 99                                                  *acter              *  
  
Name and Data Type. . . . . : HVTABLE.TIMESTMP CHARACTER(26)  
At Address. . . . . . . . . : 000869C9  
Data Value. . . . . . . . . : 2002-09-30-10.44.59.000001
```

Figure 66. Sample DB2 Information display (Part 2 of 3)

## Primary option: DB2 Information

DB2 Control Blocks							
SQL Communications Area (SQLCA) for Event # 7 Program DAPNDH11 at Address 1747E9F8 :							
Offset	Field	Name	Value	EBCDIC / ASCII			
Dec	Hex		Hex				
0	(0)	SQLCAID	E2D8D3C3 C1404040	*SQLCA	*		
8	(8)	SQLCABC	00000088	*...h	*		
12	(C)	SQLCODE	00000064	*....	*		
			<u>SQLCODE 100 Explanation</u>				
16	(10)	SQLERRML	0000	*..	*		
18	(12)	SQLERRMC	40404040 40404040 40404040 40404040	*.	*		
34	(22)		40404040 40404040 40404040 40404040	*	*		
50	(32)		40404040 40404040 40404040 40404040	*	*		
66	(42)		40404040 40404040 40404040 40404040	*	*		
82	(52)		40404040 4040	*	*		
88	(58)	SQLERRP	C4E2D5E7 D9C6D540	*DSNXRFN	*		
96	(60)	SQLERRD	FFFFFFFFFF 00000000 00000000 FFFFFFFF	*...k.....*			
112	(70)		00000000 00000000	*.....	*		
120	(78)	SQLWARN	40404040 40404040 404040	*	*		
131	(83)	SQLSTATE	F0F2F0F0 F0	*02000	*		
			<u>SQLSTATE 02000 Explanation</u>				
SQL Communications Area (SQLCA) for subsystem DB31 not shown as it is identical to the SQLCA for event # 7 program DAPNDH11.							
*** Bottom of data.							
F1=Help		F3=Exit	F5=RptFind	F6=Actions	F7=Up		
F10=Left		F11=Right			F8=Down		

Figure 67. Sample DB2 Information display (Part 3 of 3)

Information for one or more DB2 subsystems are provided. If the SQLCA data area for a DB2 subsystem is identical to one available from an Event Details display, then only a reference to the event is provided here, as shown in the example above. Otherwise, the contents of the SQLCA are shown in this display.

## Primary option: IMS Information

Selecting the “IMS Information” option provides you with a display of information that is related to IMS. There is an example display in Figure 68.

File	View	Services	Help
IMS Information			Line 1 Col 1 80
Command ==>			Scroll ==> CSR
JOBNAME: IBCB0030	USER ABEND: 4036	MVS2	2002/11/29 13:51:55
<b>Summary Detail - JOBNAME: IDCBO070</b>	SYSTEM ABEND: 0CB		FAE1 2015
IMS Version . . . . .	: V13R1M0		
IMS Region Type . . . . .	: Batch Message Processing Region		
IMS Subsystem Name. . . . .	: IB81		
Application Program Name. . .	: IDCBO070		
PSB Name. . . . .	: DFHSAM25		

Figure 68. Sample IMS Information display (Part 1 of 3)

## Primary option: IMS Information

### Last DL/I Call Parameter List

Note that storage addressed by individual parameters might no longer be valid.

Parameter 1 . . . . . : **0002BD68**  
DL/I Call Function. . . . : GU (Get Unique)

Parameter 2 . . . . . : **0001A098**  
(See "IMS Control Blocks" for details of this PCB)

Parameter 3 . . . . . : **0002BCA8**

Parameter 4 . . . . . : **8002BD48**  
SSA # 1 . . . . . : PARTROOT( = )

### IMS Control Blocks

#### Input/Output Program Communications Blocks (IOPCBs)

##### IOPCB:

At Address. . . . . : **0001A020**  
PCB Name. . . . . : IOPCB  
Relative PCB Number . . . . : 1  
PCB Type. . . . . : I/O  
Logical Terminal ID . . . . : n/a  
Status Code . . . . . : ' ' (Normal status)  
User ID . . . . . : DFHSAM25  
Group Name. . . . . : n/a  
Formatting Module Name. . . . : n/a

#### Data Base Program Communications Blocks (DBPCBs)

DBPCB (\*\* Current/Last Used \*\*):  
At Address. . . . . : **0001A098**  
PCB Name. . . . . : n/a  
Relative PCB Number . . . . : 2  
PCB Type. . . . . : Data Base or Online  
Data Base Name. . . . . : DI21PART  
Segment Level . . . . . : 01  
Status Code . . . . . : ' ' (Normal status)  
Processing Options. . . . . : A  
Segment Name. . . . . : PARTROOT  
Number of Segments. . . . . : 5  
Key Feedback Length . . . . . : 17

##### Key Feedback Data:

Address	Offset	Hex	EBCDIC / ASCII	
<b>0001A0F4</b>		<b>F0F2C1D5 F9F6F0C3 F1F04040 40404040</b>	<b>*02AN960C10</b>	*
<b>0001A104</b>	+10	<b>40</b>	*	*

#### JCB DL/I Call Trace (Most recent call first):

Call	Status			
#	Code	Description	Code	Description
1	01	GHU or GU	' '	Status good.
2	03	GHN or GN	<b>AB</b>	Segment I/O area required; none specified in call. Only applies to full-function DEQ calls.
			-or-	
	<b>BB</b>			Call could not be completed because data was unavailable and updates are backed out only since the last commit point.
			-or-	
3	03	GHN or GN	' '	End of database.
4	03	GHN or GN	' '	Status good.
5	03	GHN or GN	' '	Status good.
6	03	GHN or GN	' '	Status good.

## IMS Accounting Information

## DL/I Data Base Calls:

```

GU Calls. . . . . : 151
GN Calls. . . . . : 10050
GNP Calls . . . . . : 0
GHU Calls . . . . . : 0
GHN Calls . . . . . : 0
GHNP Calls. . . . . : 0
ISRT Calls. . . . . : 0
DLET Calls. . . . . : 0
REPL Calls. . . . . : 0
Total Calls . . . . . : 10201

```

## DL/I Message Calls:

```

GU Calls. . . . . : 0
GN Calls. . . . . : 0
ISRT Calls. . . . . : 0
PURG Calls. . . . . : 0
CMD Calls . . . . . : 0
GCMD Calls. . . . . : 0
CHNG Calls. . . . . : 0
AUTH Calls. . . . . : 0
SETO Calls. . . . . : 0

```

## DL/I System Service Calls:

```

APSB Calls. . . . . : 0
DPSB Calls. . . . . : 0
GMSG Calls. . . . . : 0
ICMD Calls. . . . . : 0
RCMD Calls. . . . . : 0
CHKP Calls. . . . . : 0
XRST Calls. . . . . : 0
ROLB Calls. . . . . : 0
ROLS Calls. . . . . : 0
SETS Calls. . . . . : 0
SETU Calls. . . . . : 0
INIT Calls. . . . . : 0
INQY Calls. . . . . : 0
LOG Calls . . . . . : 0
DB DEQ Calls. . . . . : 0
VSAM I/O Count (READs) . . : 1800
VSAM I/O Count (WRITEs) . . : 0
OSAM I/O Count (READs) . . : 0
OSAM I/O Count (WRITEs) . . : 0

```

Total DL/I I/O Count  
 (OSAM+VSAM) . . . . . : 1800

```

Total ESAF Calls. . . . . : 0
FP FLD Calls. . . . . : 0
FP POS Calls. . . . . : 0
RLSE Calls. . . . . : 0
SAVE Calls (XQUERY) . . . . : 0
RSTR Calls (XQUERY) . . . . : 0
COPY Calls (XQUERY) . . . . : 0
DL/I ICAL Calls . . . . . : 0

```

## IMS Parameter Modules

Module DFSPRPX0 Address . . . : **000078B0**

\*\*\* Bottom of data.

F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
 F10=Left F11=Right

Figure 70. Sample IMS Information display (Part 3 of 3)

## Primary option: IMS Information

The IMS Information display provides:

- General information about the IMS region
- Last DL/I call parameter list
- Information about all PCBs

All PCBs are shown in the order of their relative PCB number with identification of current (or most recently used) PCBs.

If available, JCB call trace information follows each data base PCB, showing the most recent call and up to five previous calls.

- IMS accounting information
- The address of the DFSPRPX0 parameter module

Selecting the address point-and-shoot field permits you to view the module storage in hex-dump format.

By default, the IMS Information display is shown in the "Detail" format. In this format, all PCBs are shown fully expanded. By instead selecting the "Summary" point-and-shoot field from the display header, PCBs are instead provided condensed in a table format, as in the example shown in Figure 71.

```
File View Services Help
IMS Information Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: IBCB0030 USER ABEND: 4036 MVS2 2002/11/29 13:51:55
Summary Detail - JOBNAME: IDCBO070 SYSTEM ABEND: OCB FAE1 2015

IMS Version . . . . . : V13R1M0
IMS Region Type . . . . . : Batch Message Processing Region
IMS Subsystem Name. . . . . : IB81
Application Program Name. . . : IDCBO070
PSB Name. . . . . . . . . : DFHSAM25

Last DL/I Call Parameter List

Note that storage addressed by individual parameters might no longer be valid.

Parameter 1 . . . . . : 0002BD68
DL/I Call Function. . . . . : GU (Get Unique)

Parameter 2 . . . . . : 0001A098
(See "IMS Control Blocks" for details of this PCB)

Parameter 3 . . . . . : 0002BCA8

Parameter 4 . . . . . : 8002BD48
SSA # 1 . . . . . : PARTROOT( = ) )

IMS Control Blocks

Input/Output Program Communications Blocks (IOPCBs)

PCB Cur PCB Terminal Status User Group Module
# Address PCB Name ID Code ID Name Name
1 0001A020 IOPCB n/a DFHSAM25 n/a n/a

Data Base Program Communications Blocks (DBPCBs)

PCB Cur PCB DataBase Status Segment Process Segment Key # of
# Address PCB Name Name Code Level Options Name Length Segments
2 0001A098 *** n/a DI21PART 01 A PARTROOT 17 5
```

Figure 71. Summary IMS Information display (Part 1 of 2)

```

IMS Accounting Information

DL/I Data Base Calls:
  GU Calls. . . . . : 151
  GN Calls. . . . . : 10050
  GNP Calls . . . . . : 0
  GHU Calls . . . . . : 0
  GHN Calls . . . . . : 0
  GHNP Calls. . . . . : 0
  ISRT Calls. . . . . : 0
  DLET Calls. . . . . : 0
  REPL Calls. . . . . : 0
  Total Calls . . . . . : 10201

DL/I Message Calls:
  GU Calls. . . . . : 0
  GN Calls. . . . . : 0
  ISRT Calls. . . . . : 0
  PURG Calls. . . . . : 0
  CMD Calls . . . . . : 0
  GCMD Calls. . . . . : 0
  CHNG Calls. . . . . : 0
  AUTH Calls. . . . . : 0
  SETO Calls. . . . . : 0

DL/I System Service Calls:
  APSB Calls. . . . . : 0
  DPSB Calls. . . . . : 0
  GMSG Calls. . . . . : 0
  ICMD Calls. . . . . : 0
  RCMD Calls. . . . . : 0
  CHKP Calls. . . . . : 0
  XRST Calls. . . . . : 0
  ROLB Calls. . . . . : 0
  ROLS Calls. . . . . : 0
  SETS Calls. . . . . : 0
  SETU Calls. . . . . : 0
  INIT Calls. . . . . : 0
  INQY Calls. . . . . : 0
  LOG Calls . . . . . : 0
  DB DEQ Calls. . . . . : 0
  VSAM I/O Count (READs). . . : 1800
  VSAM I/O Count (WRITEs) . . . : 0
  OSAM I/O Count (READs). . . : 0
  OSAM I/O Count (WRITEs) . . . : 0
  Total DL/I I/O Count
    (OSAM+VSAM) . . . . . : 1800
  Total ESAF Calls. . . . . : 0
  FP FLD Calls. . . . . : 0
  FP POS Calls. . . . . : 0
  RLSE Calls. . . . . : 0
  SAVE Calls (XQUERY) . . . : 0
  RSTR Calls (XQUERY) . . . : 0
  COPY Calls (XQUERY) . . . : 0
  DL/I ICAL Calls . . . . . : 0

IMS Parameter Modules

Module DFSPRDX0 Address . . . : 000078B0

*** Bottom of data.

```

Figure 72. Summary IMS Information display (Part 2 of 2)

From this display, individual PCBs can be selected by placing the cursor on the PCB # point-and-shoot field, and pressing Enter. A sample detailed PCB display is shown in Figure 73 on page 136.

## Primary option: IMS Information

```
File View Services Help
IMS Input/Output Program Communications Block (IOPCB).. Line 1 Col 1 80
Command ===> Scroll ==> CSR
JOBNAME: IDCBO070 SYSTEM ABEND: OCB FAE1 2015/01/19 14:17:15

IOPCB:
At Address. . . . . : 0001A020
PCB Name. . . . . : IOPCB
Relative PCB Number . . . : 1
PCB Type. . . . . : I/O
Logical Terminal ID . . . : n/a
Status Code . . . . . : ' ' (Normal status)
User ID . . . . . : DFHSAM25
Group Name. . . . . : n/a
Formatting Module Name. . : n/a

*** Bottom of data.
```

Figure 73. Detailed PCB display

Selecting the address point-and-shoot field from this display results in the PCB data area being shown in the Dump Storage display, as in the example shown in Figure 89 on page 149.

## Primary option: Storage Areas

Selecting the “Storage Areas” option provides you with links to any event-related formatted storage areas, any hex-dumped storage ranges, and in case of COBOL, information about any static storage for programs that are no longer on the DSA chain, as the example shown in Figure 74.

```
File View Services Help
System-Wide Storage Areas Line 1 Col 1 80
Command ===> Scroll ==> CSR
TRANID: CD01 CICS ABEND: DSNC CICS41 2003/04/28 13:22:19

Select one of the following:
1. Event 1 Program COBMST3 Storage Areas
2. Hex-Dumped Storage

The following list of COBOL programs have been called and returned during the
current execution, but do not have active register save areas:
3. Module COBMST3 Program COBFIL2 Static Storage
4. Module COBMST3 Program COBSUB2 Static Storage

*** Bottom of data.
```

Figure 74. Sample System-Wide Storage Areas display

Select any event-related point-and-shoot links from this display (such as the "Event 4 Program COBMAIN Storage Areas" link above), by one of these actions:

- Enter the option number on the command line.
- Place the cursor on the option number point-and-shoot field and press Enter.

## Primary option: Storage Areas

The result is a display that is similar to the one presented if the "Associated Storage Areas" link is selected from the detailed section for the event. See "Displaying associated storage areas" on page 140 for more information and example.

The "Hex-Dumped Storage" link provides a display of relevant unformatted storage areas which might be for one or more events. An example of the hex-dumped storage display is shown in Figure 75.

The screenshot shows a window titled "Hex-Dumped Storage". The menu bar includes "File", "View", "Services", and "Help". The title bar has "Hex-Dumped Storage" on the left and "Line 1 Col 1 80" on the right. Below the title bar, there's a command line "Command ==>". The main area displays memory dump details:

Address	Offset	Hex	EBCDIC / ASCII
Event 4 Program CDCB0010 BLL=0001 (Address 001410D0)			
See <b>Event 4 Program COBMAIN Storage Areas</b> for address range			
001410D0-001420CF formatted storage			
Event 4 Program CDCB0010 GPR 12 (Address 18206138)			
18206138		00000800 00000000 * .....	*
18206140	+8	18206C90 18207C90 00000000 00000000 *...%...@.....*	*
18206150	+18	00000000 00000000 00000000 00000000 *.....*	*
Lines 18206160-18206190 same as above			
182061A0	+68	00000000 00000000 00000000 80028B20 *.....*	*
182061B0	+78	00000000 00000000 00000000 00000000 *.....*	*
Lines 182061C0-18206240 same as above			
18206250	+118	00000000 00000000 18203FF0 00000000 *.....0....*	*
18206260	+128	00000000 00000000 00000000 00000000 *.....*	*
Lines 18206270-182062C0 same as above			
F1=Help	F3=Exit	F5=RptFind	F6=Actions F7=Up F8=Down
F10=Left	F11=Right		

Figure 75. Sample Hex-Dumped Storage display

## Primary option: Java Information

See Chapter 8, "Performing Java analysis," on page 189.

## Primary option: Language Environment Heap Analysis

Selecting the "Language Environment Heap Analysis" option provides you with a display of information that is related to the LE heap as the example shown in Figure 76 on page 138.

## Primary option: MTRACE Records

File View Services Help

---

Language Environment Heap Analysis Line 1 Col 1 80  
Command ==> \_\_\_\_\_ Scroll ==> CSR

JOBNAME: ANDYMELX ABEND: n/a FAE2 2004/08/11 14:30:35

Enclave-Level Storage  
Management (ENSM) Address : **175D97F8**

Heap allocation  
initialization value  
specified . . . . . : No

Heap free initialization  
value specified . . . . . : No

User Heap Analysis

Heap Anchor Node (HANC) . . . : **1760C000**  
Heapid. . . . . . . . . : 00000000  
Root Address. . . . . . . : **17629FA0**  
Segment Length. . . . . . . : 00020000  
Root Length . . . . . . . : 00002060  
F1=Help F3=Exit F4=Dsect F5=RptFind F6=Actions F7=Up  
F8=Down F10=Left F11=Right F12=retrieve

*Figure 76. Sample Language Environment Heap Analysis display*

## **Primary option: MTRACE Records**

Selecting the "MTRACE Records" option provides you with a display showing the MVS master trace, as the example shown in Figure 77.

File View Services Help  
 MTRACE Records Line 1 Col 1 80  
 Command ==> Scroll ==> CSR  
 SLIP DUMP ID=F092 FAE1 2006/08/21 18:11:11  
 M 0100000 FAE1 06233 18:10:40.30 STC21507 00000090 IST530I AM GBIND PENDI  
 D 016 00000090 IST1051I EVENT CODE =  
 D 016 00000090 IST1062I EVENT ID = 00  
 E 016 00000090 IST314I END  
 M 0100000 FAE1 06233 18:10:40.30 STC21507 00000090 IST530I AM GBIND PENDI  
 D 017 00000090 IST1051I EVENT CODE =  
 D 017 00000090 IST1062I EVENT ID = 00  
 E 017 00000090 IST314I END  
 N 4000000 FAE1 06233 18:10:40.63 JOB08017 00000090 IEF677I WARNING MESSAG  
 N 0020000 FAE1 06233 18:10:40.67 JOB08017 00000090 ICH7001I ANDYMEI LAS  
 N 4000000 FAE1 06233 18:10:40.68 JOB08017 00000090 \$HASP373 DACBB012 STAR  
 N 0000000 FAE1 06233 18:10:40.71 JOB08017 00000090 IEF403I DACBB012 - STA  
 N 0004000 FAE1 06233 18:10:40.99 JOB08017 00000290 -  
 -PAGING COUNTS--  
 N 0004000 FAE1 06233 18:10:40.99 JOB08017 00000290 -JOBNAME STEPNAME PRO  
 AGE SWAP VIO SWAPS STEPNO  
 N 0004000 FAE1 06233 18:10:41.01 JOB08017 00000290 -DACBB012 V00  
 0 0 0 0 1  
 N 0004000 FAE1 06233 18:10:41.25 JOB08017 00000290 -DACBB012 D00

*Figure 77. Sample MTRACE Records display*

Note that the MTRACE Records display is only available if performing analysis of an MVS dump which contains the required storage areas.

## Primary option: MTRACE Records

By placing the cursor on one of the job ID point-and-shoot fields, and pressing Enter, an MTRACE display containing only entries for the selected job IDs is shown. To return to the complete MTRACE, press PF3.

## Primary option: Abend Job Information

Selecting the "Abend Job Information" option from the initial interactive report display results in the display of the "Abend Job Information" section of the report as the example shown in Figure 78.

```
File View Services Help
Abend Job Information Line 1 Col 1 80
Command ==> TRANID: FRED Scroll ==> CSR
CICS ABEND: AEIL 2002/05/24 13:49:18

IBM Fault Analyzer Abend Job Information:

Abend Date. . . . . : 2002/05/24
Abend Time. . . . . : 13:49:18
System Name . . . . . : n/a
Job Type. . . . . . . : CICS Transaction
Job ID. . . . . . . . : STC01869
Job Name. . . . . . . : CICS04
Job Step Name . . . . . : CICS04
ASID. . . . . . . . . : 33
Job Execution Class . . . : n/a
Region Size . . . . . : 4M
EXEC Program Name . . . : DFHSIP
User ID . . . . . . . : CICSUSER
Accounting Information. . : n/a

Data Sets:

DDname Data Set or Path Name
```

Figure 78. Sample Abend Job Information display

This display provides information about the environment that existed when the fault was analyzed in real time. The information that is shown depends on the type of fault analyzed.

## Primary option: User Notes

Selecting the "User Notes" option from the initial interactive report is equivalent to issuing the NOTELIST command (for details, see "NOTELIST" on page 71), and results in the User Note List display being shown (see Figure 90 on page 153).

The "User Notes" option is dynamically added to the Interactive Reanalysis Report display whenever one or more user notes exist in a given fault entry.

## Primary option: Fault Analyzer Options

Selecting the "Fault Analyzer Options" option from the initial interactive report display results in the display of the "Fault Analyzer Options" section of the report as the example shown in Figure 79 on page 140.

## Displaying associated storage areas

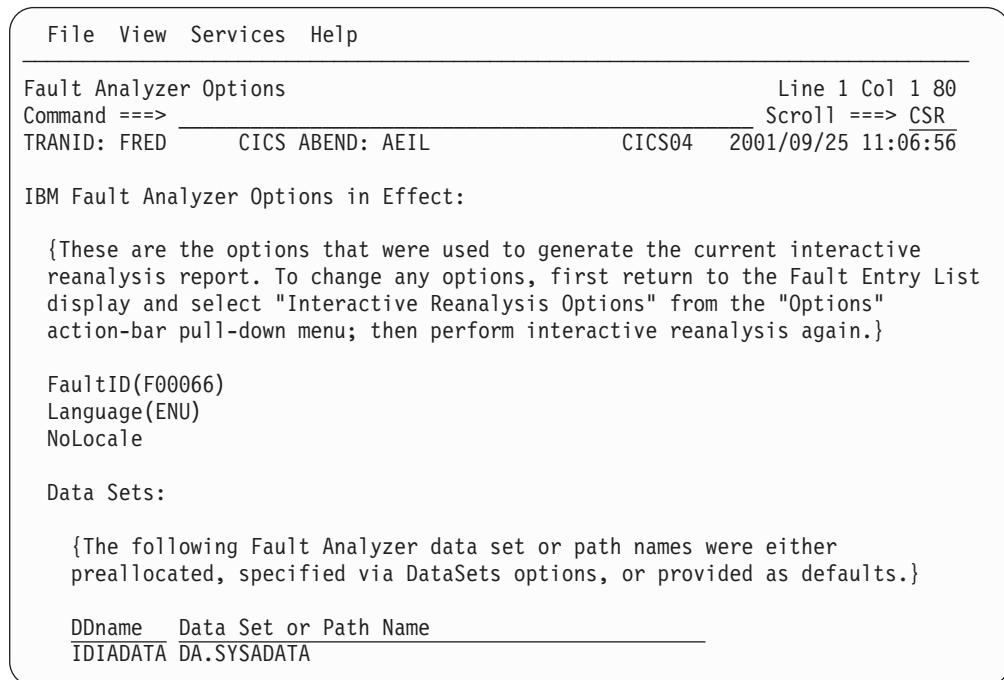


Figure 79. Sample Fault Analyzer Options display

## Displaying associated storage areas

In the interactive report, the link “Associated Storage Areas” is provided in the detailed event display depending on the programming language used. What is displayed when selecting this link depends on both the programming language and whether or not a compiler listing or side file is available to Fault Analyzer for the program:

- For COBOL programs without a compiler listing or side file, the TGT and base locators are displayed in hexadecimal dump format.
- For COBOL programs where a compiler listing or side file is supplied, the source declaration of all fields along with their current content is provided.

An example of the associated storage areas display for a COBOL program with source listing provided is shown in Figure 80 on page 141.

File View Services Help			
Associated Storage Areas		Line 1 Col 1 80 Scroll ==> CSR	
Command ==>		FAE1	2011/06/24 14:22:45
JOBNAME: COBLVL88 SYSTEM ABEND: OCB			
<b>Task Global Table (TGT) at address 17096448 for length 344</b>			
WORKING-STORAGE SECTION			
- Collapse hex - Collapse level 88			
Off	Hex Value	Data Value	Source (Starting a
<b>BLW=0000 at address 170BA0D0</b>			
0	E6D6D9D2 C9D5C760 E2E3D6D9 C1C7C540	*WORKING-STORAGE *	01 FILLER
10	40404040	*	*
<b>Suppressed Copybook</b>			
2E8	00000000	*....	* 01 FIELD-1
2F0	00000000	0	01 FIELD-2
2F8	00000000 0000	*.....	* 01 FIELD-3
			01 TABLE-4.
			03 TABLE-8 0
			04 TABLE-8A
300	426F1C29	1.111100e+02	05 ELEMENT-
304	00000000	*....	* 05 ELEMENT-
			88 JACK
			88 JILL
			ELEMENT-
314	00000000	*....	* 05 ELEMENT-

Figure 80. Sample Associated Storage Areas display

By scrolling down through the displayed information, you also find Linkage Section information, File Section information, and so on, as appropriate for the current program.

Some features are unique to the interactive reanalysis report:

- The ability to hide the hex-value column.
- The ability to collapse level 88 items.
- The ability to show all COBOL base locators.

These are explained in the following topics.

## Hiding the hex-value column

Hiding the hex-value column might be useful for users of narrow (80-column) screens. By placing the cursor on the minus sign above the "Hex Value" heading in Figure 80 and pressing Enter, the display is changed to that shown in Figure 81 on page 142:

## Displaying associated storage areas

```
File View Services Help
Event 1 Program COBLVL88 Storage Areas Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBLVL88 SYSTEM ABEND: 0CB FAE1 2011/06/24 14:22:45
Task Global Table (TGT) at address 17096448 for length 344
WORKING-STORAGE SECTION
+ Expand hex - Collapse level 88
Off Data Value Source (Starting at Line # 000010 )
BLW=0000 at address 170BA0D0
 0 *WORKING-STORAGE * 01 FILLER          PIC X(20)  VALUE 'WORKING-
10 *           *
Suppressed Copybook
2E8 *....      * 01 FIELD-1          PIC 999999 COMP-3.
2F0 0          01 FIELD-2          PIC 999999 COMP-4.
2F8 *.....     * 01 FIELD-3          PIC 999999.
          01 TABLE-4.
          03 TABLE-8 OCCURS 6 TIMES.
          04 TABLE-8A OCCURS 3 TIMES.
          05 ELEMENT-1          COMP-1.
300 1.111100e+02 * 05 ELEMENT-2 OCCURS 4 TIMES          PIC XXXX
304 *....       *          88 JACK VALUE 'JACK'.
          88 JILL VALUE 'JILL'.
          ELEMENT-2(1,1,2) to ELEMENT-2(1,1,4) same as
314 *....       * 05 ELEMENT-3          PIC 999999 COMP-3
```

Figure 81. Sample Associated Storage Areas display with hex value column collapsed

Note that the minus sign point-and-shoot field, now above the "Data Value" column, has changed to a plus sign, and that if placing the cursor on this point-and-shoot field, the display changes back to the first display with "Hex Value" visible.

The last selected "Collapse/Expand hex" option is saved in the user's ISPF profile and is used as the default display mode on subsequent Associated Storage Areas displays.

## Collapsing level 88 items

Collapsing the level 88 items can be used to suppress, potentially many and not very useful declarations, from the display. By placing the cursor on the minus sign immediately ahead of the "Collapse level 88" heading in Figure 80 on page 141 and pressing Enter, the display is changed to that shown in Figure 82 on page 143:

## Displaying associated storage areas

```

File View Services Help
Event 1 Program COBLVL88 Storage Areas Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBLVL88 SYSTEM ABEND: OCB FAE1 2011/06/24 14:22:45

Task Global Table (TGT) at address 17096448 for length 344

WORKING-STORAGE SECTION
- Collapse hex + Expand level 88
Off Hex Value Data Value Source (Starting a
BLW=0000 at address 170BA0D0
  0 E6D6D9D2 C9D5C760 E2E3D6D9 C1C7C540 *WORKING-STORAGE * 01 FILLER
  10 40404040          *           *
Suppressed Copybook
  2E8 00000000      *....      * 01 FIELD-1
  2F0 00000000      0          01 FIELD-2
  2F8 00000000 0000 *.....      * 01 FIELD-3
                                01 TABLE-4.
                                03 TABLE-8 0
                                04 TABLE-8A
  300 426F1C29      1.111100e+02    05 ELEMENT-
  304 00000000      *....      * 05 ELEMENT-
Level 88 Items
                                ELEMENT-
  314 00000000      *....      * 05 ELEMENT-
  318 00000000      0          05 ELEMENT-

```

Figure 82. Sample Associated Storage Areas display with level 88 items collapsed

Note that the minus sign point-and-shoot field, now immediately ahead of the "Expand level 88" heading, has changed to a plus sign, and that if placing the cursor on this point-and-shoot field, the display changes back to the first display with level 88 items inlined.

Placing the cursor on the "Level 88 Items" point-and-shoot field, and pressing Enter, results in the display shown in Figure 83.

```

File View Services Help
Level 88 Items Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: COBLVL88 SYSTEM ABEND: OCB FAE1 2011/06/24 14:22:45

Source
  88 JACK VALUE 'JACK'.
  88 JILL VALUE 'JILL'.

*** Bottom of data.

```

Figure 83. Sample level 88 items display

## Displaying associated storage areas

Press PF3 to return to the associated storage areas display.

The last selected "Collapse/Expand level 88" option is saved in the user's ISPF profile and is used as the default display mode on subsequent Associated Storage Areas displays.

## Showing all COBOL base locators

If source code is not available for an event, and multiple contiguous base locators exist of the same storage type (for example, WORKING-STORAGE), then these are initially displayed summarized. In this case, the starting address is shown and the length is the total length of all the contiguous base locators that follow.

Place the cursor on the plus sign that precedes "Show all BLs", and press Enter. The display changes to show all base locators separately with each their own starting address and length. At the same time, the plus sign at the top of the display changes to a minus sign, and the text changes to "Summarize BLs".

To switch back to the default summarized base locator display, place the cursor on the minus sign that precedes "Summarize BLs", and press Enter.

## Expanding messages and abend codes

Messages or abend codes are initially never expanded when using the interactive dump reanalysis feature of Fault Analyzer. This lack of expansion is to prevent the need to scroll through potentially very long explanations to see report items that might follow. Instead, to view the explanation for messages or abend codes in the interactive report, one can place the cursor on the message identifier or abend code and press the Enter key. This opens a display similar to what you see in the batch report as the example shown in Figure 84.

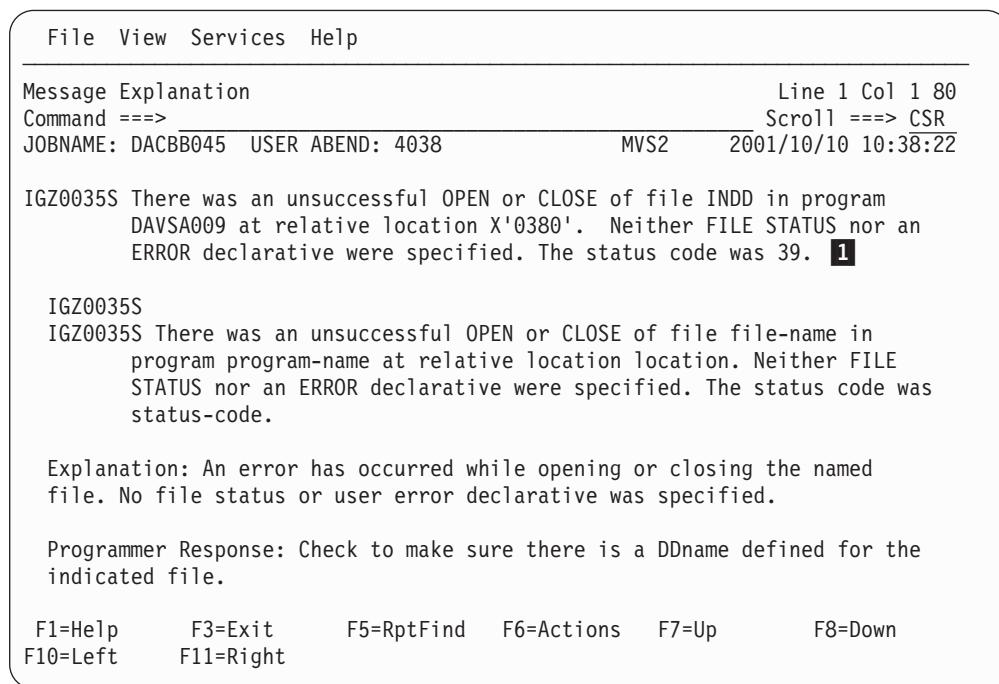


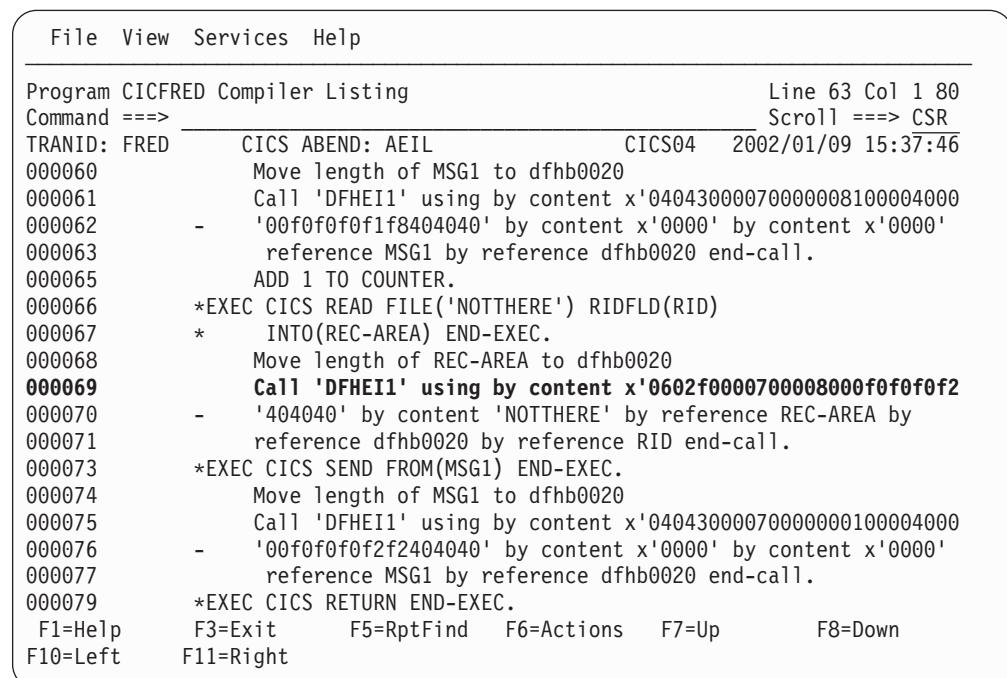
Figure 84. Sample Message Explanation display

When a message is displayed in the report, the first occurrence of the message contains the actual text from when it was issued (see **1** in the above example). If the instance-specific text is not available, then a comment “job-specific text not available” replaces it.

In the expansion that immediately follows the issued message or abend code is its explanation that is generally obtained from the softcopy books provided with Fault Analyzer. If an explanation for the message or abend code cannot be found, then the text “explanation not available” replaces it.

## Displaying source code

To display the source code for an entire program, place the cursor on any yellow point-and-shoot source line number or listing statement number and press Enter. For example, if line number 66 was selected from an event in the interactive report, the display that is shown in Figure 85 might be displayed.



```

File View Services Help
Program CICFRED Compiler Listing Line 63 Col 1 80
Command ==> Scroll ==> CSR
TRANID: FRED CICS ABEND: AEIL CICS04 2002/01/09 15:37:46
000060 Move length of MSG1 to dfhb0020
000061 Call 'DFHEI1' using by content x'04043000070000008100004000
000062 - '00f0f0f0f1f8404040' by content x'0000' by content x'0000'
000063 reference MSG1 by reference dfhb0020 end-call.
000065 ADD 1 TO COUNTER.
000066 *EXEC CICS READ FILE('NOTTHERE') RIDFLD(RID)
000067 * INTO(REC-AREA) END-EXEC.
000068 Move length of REC-AREA to dfhb0020
000069 Call 'DFHEI1' using by content x'0602f0000700008000f0f0f0f2
000070 - '404040' by content 'NOTTHERE' by reference REC-AREA by
000071 reference dfhb0020 by reference RID end-call.
000073 *EXEC CICS SEND FROM(MSG1) END-EXEC.
000074 Move length of MSG1 to dfhb0020
000075 Call 'DFHEI1' using by content x'0404300007000000100004000
000076 - '00f0f0f0f2f2404040' by content x'0000' by content x'0000'
000077 reference MSG1 by reference dfhb0020 end-call.
000079 *EXEC CICS RETURN END-EXEC.
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down
F10=Left F11=Right

```

Figure 85. Sample Compiler Listing display

The source line or statement number that is initially selected is highlighted.

The example that is shown here assumes that the display of pseudo-assembler instructions is suppressed. For details on this suppression, see “Displaying source code.”

Information that is displayed in blue (all lines that start in column 1) pertains to the program source code. On the left side of the display is information about the source line or listing statement number of the source displayed. This information is followed by the actual source code at this location in the program.

To add machine instruction information to the listing, select the **View** menu Add Pseudo Assembler Instructions option (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This option causes the Compiler Listing display to be reformatted with pseudo-assembler instructions

## Displaying source code

inserted into the program source code as shown in Figure 86.

The screenshot shows a window titled "Program CICFRED Compiler Listing". The top status bar indicates "Line 316 Col 1 80" and "Scroll ==> CSR". Below the title, it says "Command ==> TRANID: FRED" and "CICS ABEND: AEIL". The date and time are listed as "CICS04 2002/01/09 15:37:46". The main area displays assembly language code. A green line at address 000069 contains the instruction "Call 'DFHEI1' using by content x'0602f0000700008000f0f0f0f2". This line is preceded by several comments and machine instructions. At the bottom of the code area, there is a series of function key definitions: F1=Help, F3=Exit, F5=RptFind, F6=Actions, F7=Up, F8=Down, F10=Left, and F11=Right.

Figure 86. Sample Compiler Listing display: Pseudo-assembler instructions enabled

Information that is displayed in green (all lines that start in column 15) pertains to the disassembly of machine instructions. This information is shown interspersed with the source code information, following the line of code to which they belong.

**Note:** For COBOL programs compiled with TEST(NONE,SYM,SEPARATE), all pseudo-assembler instructions are placed following the last line of source code.

For all other programs, ensure that the source line of interest is scrolled to the top of the display before you add pseudo-assembler instructions to the listing display. The extra display lines might cause the source line to disappear from the current view.

To remove the pseudo-assembler instructions from the display, select the **View** menu Remove Pseudo Assembler Instructions option.

Information at the top of the listing indicates the source of the compiler listing or side. An example is shown in Figure 87 on page 147. You can scroll to the top of the side file listing by entering, for example, the UP MAX command.

The screenshot shows a window titled "Program CICFRED Compiler Listing". The top right corner displays "Line 1 Col 1 80", "Scroll ==> CSR", and the date/time "CICS04 2002/01/09 15:37:46". Below this, a message states: "The listing or side file used for the following was found in DA.LISTING.COBOL(CICFRED).". The main area is labeled "Source Line #". It lists several lines of COBOL code, starting with the Identification Division and followed by Environment, Data, and Working-STORAGE sections. At the bottom of the listing, there are function key definitions: F1=Help, F3=Exit, F5=RptFind, F6=Actions, F7=Up, F8=Down, F10=Left, and F11=Right.

Figure 87. Sample Compiler Listing display: Origin information

## Displaying storage locations

To display storage locations, place the cursor on any yellow point-and-shoot address (for example, a register value), and press the Enter key. Alternatively, use the SHOW command (see "SHOW" on page 73 for details).

An example of the storage display is shown in Figure 88.

The screenshot shows a window titled "Dump Storage". The top right corner displays "17C01380-17C013D7", "Scroll ==> CSR", and the date/time "MVS2 2003/08/12 13:46:58". Below this, a message states: "JOBNAME: IDIVPCOB SYSTEM ABEND: OC7". The main area is a table with columns "Address", "Offset", "Hex", and "EBCDIC / ASCII". It lists memory dump data for various addresses, including module headers and specific memory locations. At the bottom of the dump, there are function key definitions: F1=Help, F3=Exit, F7=Up, F8=Down, F10=Prev, and F11=Next.

Figure 88. Sample Dump Storage display

## Displaying storage locations

The character-interpreted section on the right-hand side of the hex data is generally displayed as EBCDIC. To instead display the data as ASCII, place the cursor on the ASCII point-and-shoot field and press Enter. With the cursor now on the EBCDIC point-and-shoot field, pressing Enter a second time reverts to the EBCDIC display.

Placing the cursor anywhere in the hexadecimal storage display area, and pressing the Enter key, takes you to the selected address. If the point-and-shoot field in which the cursor is placed is fewer than 8 digits, then it is padded with leading zeroes to form an 8-digit 31-bit address.

Overtyping the first two digits of an 8-digit address point-and-shoot field with zeroes, immediately prior to pressing the Enter key, ensures that the address is interpreted as a 24-bit address.

A 64-bit address is selected by overtaking the last digit of the point-and-shoot field which represent the first half of the 64-bit address (bits 0-31), or by overtaking the first digit of the point-and-shoot field which represent the second half of the 64-bit address (bits 32-63), before pressing Enter. This process logically 'joins' the point-and-shoot field closest to the underscore with the field in which it is placed. For example, given the following two adjacent address point-and-shoot fields

**00000001 80109020**

and either overtaking the last digit of the first field like this

**00000001 80109020**

or the first digit of the second field like this

**00000001 0109020**

results in the 64-bit address 00000001\_80109020 being displayed. As with 31-bit addresses, the second half of the 64-bit address is padded with leading zeroes to 8-digits.

Given the following

A record is maintained of the last 10 addresses displayed. You can use the PREV (PF10) or NEXT (PF11) commands to redisplay areas previously selected.

The number of bytes per line shown depends on the visible width, not the current preferred formatting width. 32 bytes per line are shown if the visible width permits, otherwise 16 bytes are shown.

If available, a description of the initially selected address is provided, along with descriptions of the beginning of other storage areas, such as modules and programs, and any user notes (see "Creating and managing user notes" on page 150).

The FIND command used from this display behaves different to that of all other displays, since it is the minidump which is searched instead of the formatted display itself. For details, see "FIND command differences between display types" on page 68.

To display storage ahead of, or following, the storage in the current display, use the UP/DOWN commands as appropriate (usually mapped to PF7/PF8).

Alternatively, an offset can be entered on the command line in the format:

**Syntax**

```
► + [hex-offset] ►
```

where *hex-offset* is a hexadecimal offset relative to the top left address shown. For example:

+10C

or

-D4

## Displaying data areas

When selecting address point-and-shoot fields from within the interactive reanalysis report, the result is generally the display of all available storage but with the selected address at offset 0. However, in some cases, where the address is associated with a data area of a given length, the resulting display initially shows only the storage for the implied length and with all storage prior to, or following, shown as "suppressed". An example of this is shown in Figure 89.

File View Services Help			
Dump Storage			17C01380-17C013D7
Command ==> ShowAll - JOBNAME: IDCB0070 SYSTEM ABEND: 0CB			Scroll ==> CSR FAE1 2015/01/19
Address	Offset	Hex	EBCDIC / ASCII
		Address range 00000000-0001A01F suppressed	
		PCB #1 - Start	
0001A020		00400038 00010018 40404040 00000000 *.. .... .....	
0001A030	+10	008A1054 00000000 12A74084 C9D6D7C3 *.....x dIOPC*	
0001A040	+20	C2404040 00000000 00000000 00000000 *B .....*	
0001A050	+30	00000000 0001A020	*..... *
		Event 5 Program IDCB0070 BLL=00001	
0001A058	+38	40404040 40404040 *	*
0001A060	+40	10004040 0000000F 0000000F 00000000 *.. .....	*
0001A070	+50	00000000 00000000 C4C6C8E2 C1D4F2F5 *.....DFHSAM25*	
0001A080	+60	40404040 40404040 00000000	* ... *
		PCB #1 - End	
		Address range 0001A08C-FFFFFFFFFF_FFFFFFFF suppressed	

Figure 89. Sample Dump Storage suppressed display

To see all available storage, place the cursor on the ShowAll point-and-shoot field in the display title, and press Enter.

Alternatively, selecting any hexadecimal address point-and-shoot field results in all available storage being shown.

If the ShowAll method is used, then there is no way possible to again suppress the surrounding storage, other than by pressing PF3 and reselecting the original address point-and-shoot field.

However, if an address is instead selected, then the address is "stacked" as per usual in the Dump Storage display, and if pressing PF10 until the original address

## Displaying storage locations

is again displayed, the surrounding storage is again be suppressed. This is the case unless more than 10 subsequent addresses have been selected.

## Creating and managing user notes

User notes are comments that the interactive user can add against any storage location. They are saved in the history file fault entry and are available to all users.

User notes are created from the displays:

- Dump Storage
- Associated Storage Areas
- Event-Related Storage Areas (Event *n* Program *name* Storage Areas)
- Hex-Dumped Storage

In the following, these are simply referred to as "storage areas" displays.

The user notes are created by placing the cursor on the area of storage to which the note applies, typing one or more characters that are distinguishable from hexadecimal digits, and pressing Enter. For example, given the Dump Storage display shown in Figure 88 on page 147, placing the cursor at the address 17C01389 storage, and typing "This is", the following display would be expected:

File View Services Help			
Dump Storage	17C01380-17C013D	Command ==>	Scroll ==> CSR
JOBNAME: IDIVPCOB	SYSTEM ABEND: OC7	MVS2	2003/08/12 13:46:58
Address	Offset	Hex	EBCDIC / ASCII
Module IDISCBL1 program IDISCBL1 + X'640', source line # 32			
Event 1 Program IDISCBL1	GPR 1 + X'1051'		
Event 1 Program IDISCBL1	GPR 10 + X'1300'		
Event 1 Program IDISCBL1	GPR 11 + X'1008'		
Event 1 Program IDISCBL1	GPR 12 + X'1348'		
Event 1 Program IDISCBL1	GPR 14 + X'680'		
Event 1 Program IDISCBL1	GPR 3 + X'600'		
Event 1 Program IDISCBL1	GPR 4 + X'1544'		
17C01380	20004160	61345060	51This i s10C310A *....-/..&..K.....*
17C01390	+10	D2035110	310E5860 20085060 5114D20B *K.....-&..K.*
17C013A0	+20	51183112	58602004 50605128 58602008 *.....-..&.....*
17C013B0	+30	5060512C	D2175130 311E98EC D00C07FE *&-..K.....q.}...*
Module IDISCBL1 CSECT CEESG005			
17C013C0	+40	E2F0F0F5	00140001 00000000 00000000 *S005.....*
17C013D0	+50	00000000	00000000 *..... *
Module IDISCBL1 CSECT CEEBETBL			
F1=Help	F3=Exit	F7=Up	F8=Down F10=Prev F11=Next

Pressing Enter results in an ISPF edit panel being presented, initialized with the text typed:

## Displaying storage locations

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
Volume: -----
EDIT      Note.17C01389          Columns 00001 00072
Command ==> _____ Scroll ==> PAGE
***** **** Top of Data ****
000001 This is
***** **** Bottom of Data ****
F1=Help      F2=Split      F3=Exit      F4=:tf       F5=Rfind      F6=Rchange
F7=Up       F8=Down       F9=Swap       F10=Left     F11=Right     F12=Cancel
```

From here the note can be completed, adding as many lines as required. The first line should be treated as a heading as it is the only line of the note shown if the display of the note is later collapsed.

Assuming that the final note is as follows:

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
Volume: -----
EDIT      Note.17C01389          Columns 00001 00072
Command ==> _____ Scroll ==> PAGE
***** **** Top of Data ****
000001 This is an important reminder!
000002 The contents of storage at this offset into this module could be
000003 significant for the understanding of the error that caused this fault.
***** **** Bottom of Data ****
F1=Help      F2=Split      F3=Exit      F4=:tf       F5=Rfind      F6=Rchange
F7=Up       F8=Down       F9=Swap       F10=Left     F11=Right     F12=Cancel
```

Press PF3 to return to the storage areas display, which shows the newly created user note, inserted immediately ahead of the storage to which it belongs.

## Displaying storage locations

File View Services Help

---

Dump Storage 17C01380-17C013D7  
 Command ==> Scroll ==> CSR  
 JOBNAME: IDIVPCOB SYSTEM ABEND: OC7 MVS2 2003/08/12 13:46:58

Address	Offset	Hex	EBCDIC / ASCII
		Module IDISCBL1 program IDISCBL1 + X'640', source line # 32	
		Event 1 Program IDISCBL1 GPR 1 + X'1051'	
		Event 1 Program IDISCBL1 GPR 10 + X'1300'	
		Event 1 Program IDISCBL1 GPR 11 + X'1008'	
		Event 1 Program IDISCBL1 GPR 12 + X'1348'	
		Event 1 Program IDISCBL1 GPR 14 + X'680'	
		Event 1 Program IDISCBL1 GPR 3 + X'600'	
		Event 1 Program IDISCBL1 GPR 4 + X'1544'	
<b>17C01380</b>		<b>20004160 61345060 51</b>	*...-/.&.. *
- This is an important reminder! The contents of storage at this offset into this module could be significant for the understanding of the error that caused this fault.			
17C01389	+9	08D203 510C310A *	.K.....*
17C01390	+10	D2035110 310E5860 20085060 5114D20B *K.....-&..K.*	
17C013A0	+20	51183112 58602004 50605128 58602008 *.....-&.....*	
17C013B0	+30	5060512C D2175130 311E98EC D00C07FE *&..K.....q.}....*	
F1=Help	F3=Exit	F7=Up	F8=Down F10=Prev F11=Next

By default, all user notes are shown "expanded", as indicated by the minus sign point-and-shoot field preceding the note. By placing the cursor on this field, and pressing Enter, the note is "collapsed" as shown below:

File View Services Help

---

Dump Storage 17C01380-17C013D7  
 Command ==> Scroll ==> CSR  
 JOBNAME: IDIVPCOB SYSTEM ABEND: OC7 MVS2 2003/08/12 13:46:58

Address	Offset	Hex	EBCDIC / ASCII
		Module IDISCBL1 program IDISCBL1 + X'640', source line # 32	
		Event 1 Program IDISCBL1 GPR 1 + X'1051'	
		Event 1 Program IDISCBL1 GPR 10 + X'1300'	
		Event 1 Program IDISCBL1 GPR 11 + X'1008'	
		Event 1 Program IDISCBL1 GPR 12 + X'1348'	
		Event 1 Program IDISCBL1 GPR 14 + X'680'	
		Event 1 Program IDISCBL1 GPR 3 + X'600'	
		Event 1 Program IDISCBL1 GPR 4 + X'1544'	
<b>17C01380</b>		<b>20004160 61345060 51</b>	*...-/.&.. *
+ This is an important reminder!			
17C01389	+9	08D203 510C310A *	.K.....*
17C01390	+10	D2035110 310E5860 20085060 5114D20B *K.....-&..K.*	
17C013A0	+20	51183112 58602004 50605128 58602008 *.....-&.....*	
17C013B0	+30	5060512C D2175130 311E98EC D00C07FE *&..K.....q.}....*	
		Module IDISCBL1 CSECT CEESG005	
17C013C0	+40	E2F0F0F5 00140001 00000000 00000000 *S005.....*	
F1=Help	F3=Exit	F7=Up	F8=Down F10=Prev F11=Next

The preceding point-and-shoot field now indicates "collapsed" by a plus sign instead. By simply placing the cursor on this point-and-shoot field, the user can toggle between the collapsed and expanded view.

It is also possible to overtype the point-and-shoot field with two more action characters (not case-sensitive):

- D** Used to delete the user note.
- E** Used to edit the user note.

To see all user notes that exist for the current fault entry, enter the NOTELIST command from the command line of any display within the interactive report, or select the "List User Notes" option from the View action-bar pull-down menu. The result is a display like the example shown in Figure 90:

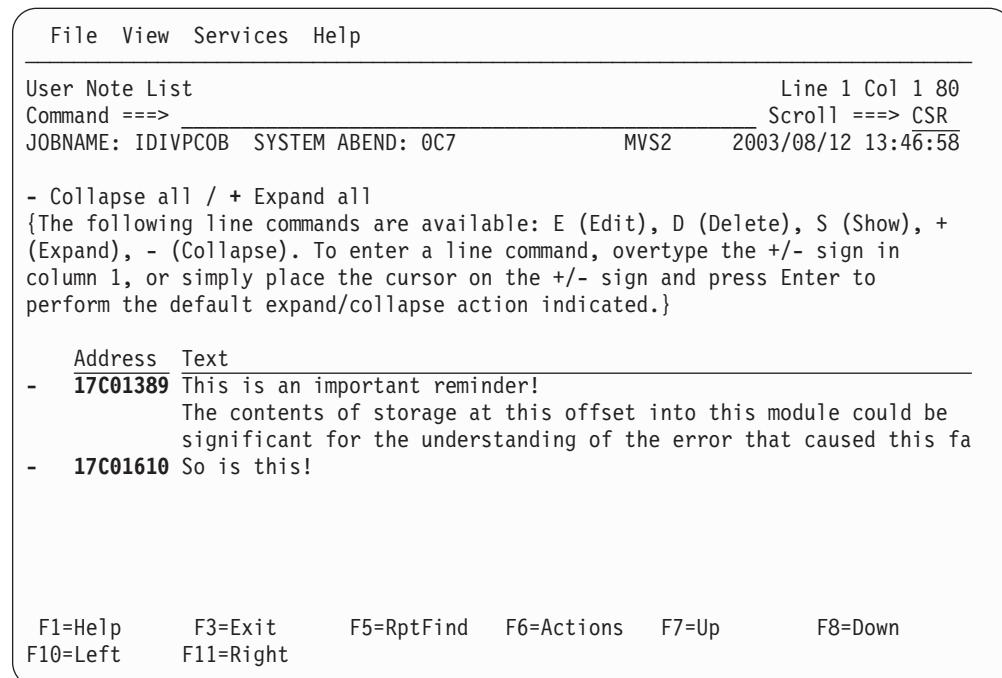


Figure 90. Sample User Note List display

As indicated in the optional help text on this display, the point-and-shoot field preceding each user note can be overtyped to request a specific action in the same way as in the storage areas displays.

Additionally, the User Note List display permits all user notes to be expanded or collapsed simultaneously by selecting the "expand all" or "collapse all" point-and-shoot fields at the top of the display. The expand/collapse state of any note is common between the User Note List display and the Dump Storage display, so that any changes made in one display is reflected in the other.

To display the storage that is associated with a user note, use the S line command, or place the cursor on the address point-and-shoot field, and press Enter.

User notes are saved in the history file fault entry when the user exits from the interactive report. At this time, if user notes have been added or modified, the user is prompted to acknowledge the update of the fault entry with a display as the example shown in Figure 91 on page 154:

## Displaying storage locations

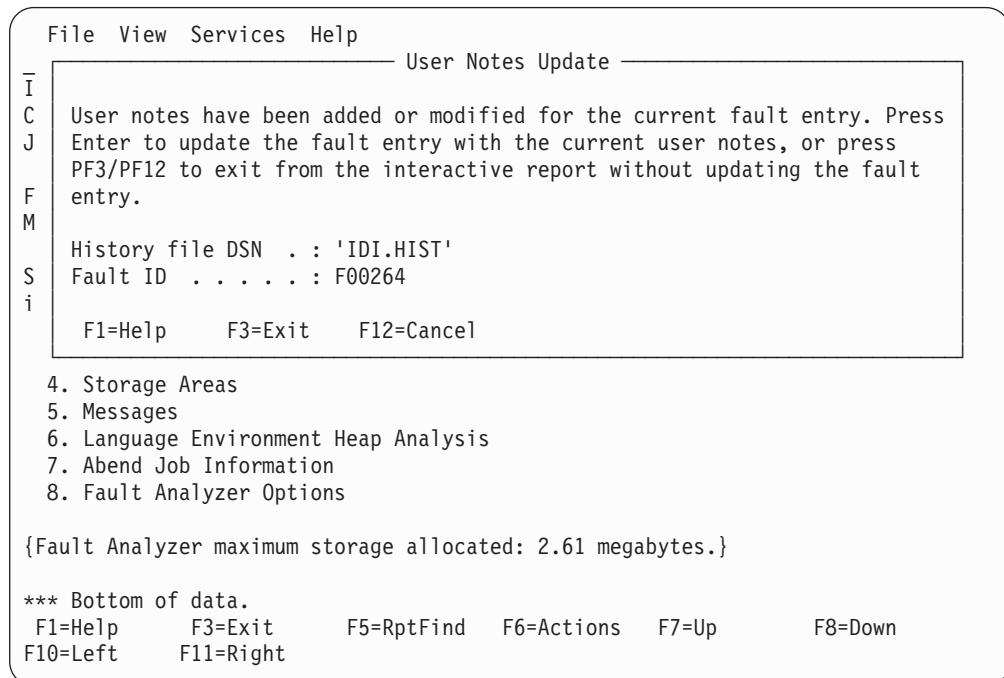


Figure 91. Sample User Notes Update prompt

Pressing Enter from this display permits the history file fault entry to proceed, while pressing PF3 or PF12 exits from the interactive report without any updates to user notes.

## Mapping storage areas using DSECT information

By using the DSECT command from within the interactive report, storage areas can be mapped based on PDS(E) data set members containing assembler macro or DSECT copybooks.

The DSECT command (see “DSECT” on page 66 for syntax), can be entered from the command line of any display, or via PF key assignment. By default, the DSECT command is assigned to PF4.

When invoked, you are shown a popup window similar to the following:

## Mapping storage areas using DSECT information

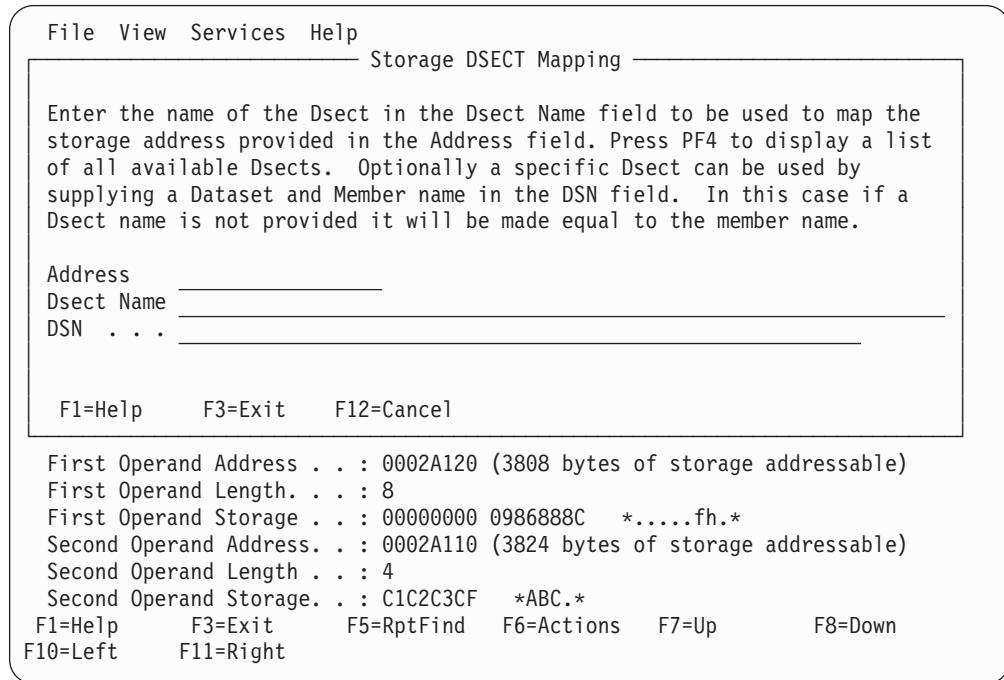


Figure 92. Sample Storage DSECT Mapping Entry display

**Note:** If the DSECT command is issued while the cursor is on an address point-and-shoot field, then the popup display address is automatically initialized to that address.

You can now supply the start address (if not already filled in), and the name of the DSECT to use when mapping the storage area. The Address field is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

The name of the DSECT can be provided in one of two ways:

1. Just the DSECT name can be entered, in which case the IDIDSECT concatenation is searched for a match (see “IDIDSECT concatenation” on page 156 for details on the IDIDSECT concatenation). If multiple occurrences of the requested DSECT exist in the IDIDSECT concatenation, then press PF4 and select the appropriate one from the resulting list of all available DSECTs. A DSECT is selected from the list using an 'S', or it can be Edited by entering an 'E'.
2. The PDS(E) data set and member name, where the specified DSECT is stored, can be supplied. If a DSECT name is not provided, then it defaults to be the same as the PDS(E) member name.

Once a valid storage address and DSECT name have been specified, then the storage is mapped and displayed, similar to the following example:

## Mapping storage areas using DSECT information

```
Dsect mapping for DFHCSADS at address 4de20
Command ==> _____ Line 1 Col 1 80
                                                               Scroll ==> CSR

CICS DUMP: SYSTEM=QXPM2C61 CODE=ASRA      ID=      MVS2      2003/06/25 13:47:55

0004DE20 +0000          DSECT DFHCSADS
                           DFHCSABA EQU   *
0004DE20 +0000 00000248 0000D0A0 17EB4D00 983C1ECE
                           80BF4DA8 80800000 18685160 18642330
                           000003FD 18973FB8 00000BAF 00000000
                           983C1A40 18973000 18685160 18684B70
                           00051D80 17F90680    CSAOSRSA DS  18F
                           CSASOSI  DS  0B
                           CSASSI1  DS  B
                           CSAFPURG EQU  X'80'
                           CSAFTCAB EQU  X'40'
                           CSASDTRN EQU  X'20'
                           CSACSDOP EQU  X'02'
                           CSASOSON EQU  X'01'
                           CSAKCMII DS  0B
                           CSASSI2  DS  B
                           F8=Down
0004DE68 +0072 00
                           F1=Help     F3=Exit     F5=RptFind   F6=Actions   F7=Up
                           F10=Left    F11=Right

F1=Help     F3=Exit     F5=RptFind   F6=Actions   F7=Up
F10=Left    F11=Right
```

Figure 93. Sample Storage DSECT Mapping Map display

Scroll up/down or right/left as needed to display more DSECT information.

Press PF3 to return from the Storage DSECT Mapping Map display.

## IDIDSECT concatenation

The IDIDSECT concatenation can optionally be specified in the DATASETS sections of your options data set. If specified, then it should specify the name of one or more PDS(E) data sets, which contain DSECT files to be used when processing the DSECT command.

The IDIDSECT data set attributes must be one of the following:

- RECFM=FB and LRECL $\geq$ 80
- RECFM=VB and LRECL $\geq$ 84

Regardless of LRECL, only the first 80 bytes of each record are read. The recommended IDIDSECT data set attributes are RECFM=FB and LRECL=80.

When in an interactive report, the first time the DSECT command is issued, it processes each data set in the IDIDSECT concatenation. If the data set contains a \$DINDEX member (see “Indexing your DSECT data sets (\$DINDEX member)” on page 157), then the DSECT details in this member are used. Otherwise, each member in the data set is assumed to contain a DSECT of the same name. Once all the DSECT details have been determined for a data set, then the process is repeated for the next data set, until all the data sets in the IDIDSECT concatenation have been processed.

Note that this process only happens once per interactive report session. If new DSECTs are added to a data set in the IDIDSECT concatenation, or if the \$DINDEX member is updated, then you must do one of these actions:

- Restart the interactive report.

- Explicitly identify the new DSECT by specifying the data set and member name it is contained in.

### Indexing your DSECT data sets (\$DINDEX member)

To allow for DSECT names of up to the maximum of 63 characters, and individual PDS(E) members containing multiple DSECTS, a \$DINDEX member can be created. The \$DINDEX member (which can be created using the IDIPDSCU utility, see “DSECT indexing utility (IDIPDSCU)”) should contain a line for every DSECT in each member of the PDS(E). The format of each line should be the DSECT name, followed by a space, followed by the member in which that DSECT is found. For example:

```
DSECT1 MEMBER1
DSECT2 MEMBER1
LONGDSECTNAME1 MEMBER2
LONGDSECTNAME2 MEMBER2
```

In this example, PDS(E) member MEMBER1 contains DSECTs DSECT1 and DSECT2, and MEMBER2 contains DSECTs LONGDSECTNAME1 and LONGDSECTNAME2.

### DSECT indexing utility (IDIPDSCU)

The IDIPDSCU utility is used to create a \$DINDEX member for a given data set (see “Indexing your DSECT data sets (\$DINDEX member)” for details about this member). It does this by creation calling the assembler for each member in the data set, and extracting the imbedded DSECTs from the assembler output.

In situations where DSECT or macro expansions require special keyword specifications, separate members might have to be coded. These members need to call the macro in question, providing the required keywords, and need to be stored in the same data set as the macro they invoke. For example, CICS provides in its SDFHMAC data set a member called DFHTCTZE, which provides multiple terminal-related DSECTs. If this member is processed directly by the IDIPDSCU utility, then it does not detect the TCTENIB DSECT, as detection requires special macro keywords to be specified. In this case, if a member is created in the SDFHMAC data set (or a copy of it), which contains the following source line, then all DSECTs are detected, including TCTENIB:

```
DFHTCTZE CICSYST=YES
```

The IDIPDSCU utility can be used either by entering IDIPDSCU next to a data set name in ISPF, or as a batch utility, in which case the data set to process is passed as a parameter. See the following example:

```
//UTILJOB1 JOB ...
//RUNUTIL EXEC PGM=IDIPDSCU,PARM='fully_qualified_PDS(E)_data_set_name'
//SYSPRINT DD SYSOUT=*
```

The IDIPDSCU utility creates a \$DINDEX member in the target data set, so you must have write access to this data set.

---

### Displaying chained data areas

Using the RUNCHAIN command (see “RUNCHAIN” on page 73, storage can be scanned for a chain of linked control blocks. The RUNCHAIN command can be invoked either by entering RUNCHAIN on any interactive report command line, or by assigning RUNCHAIN to a PF key. When invoked, you are shown a popup

## Displaying chained data areas

panel similar to the following:

The screenshot shows a command-line interface for the Storage RUNCHAIN Command. The menu bar includes File, View, Services, Help, and the command name. A sub-menu bar below 'Services' lists Storage, RUNCHAIN, and Command. The main area contains the following text:

Enter the required fields and press Enter .

Start Address . . . . .  
Max Number Control Blocks 9999 (Decimal)  
Forward Pointer Offset . . . (Hex)  
End of Chain Identifier . . . (Hex, Default Values: All 0's and all F's)  
Eyecatcher Text . . . . .  
Eyecatcher Offset . . . . . (Hex)

Function keys at the bottom: F1=Help F3=Exit F12=Cancel

Output from the command:

```
000011 n/a      01 PARM1          PIC X(4) .
000012 n/a      01 PARM2.
```

Data Field Values:

```
PARM1 = 0001
PARM2 = HEADING FOR IDIXSNAP
F1=Help   F3=Exit   F5=RptFind   F6=Actions   F7=Up       F8=Down
F10=Left  F11=Right
```

Figure 94. Sample Storage RUNCHAIN Command entry display

The Start Address is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

If a 31-bit Start Address is specified:

- The forward pointer at the Forward Pointer Offset is assumed to be a 31-bit address.
- The End of Chain Identifier is assumed to be a 32-bit hexadecimal value. If less than 8 hexadecimal digits are specified, then the specified value is padded with leading zeroes to form a 32-bit hexadecimal value.

If a 64-bit Start Address is specified:

- The forward pointer at the Forward Pointer Offset is assumed to be a 64-bit address.
- The End of Chain Identifier is assumed to be a 64-bit hexadecimal value. If less than 16 hexadecimal digits are specified, then the specified value is padded with leading zeroes to form a 64-bit hexadecimal value.

For a given Start Address and Forward Pointer Offset, the RUNCHAIN command follows the chain of control blocks until one of the following end conditions is met:

1. The number of control blocks scanned has exceeded the maximum number set by the user (the default value is 9999).
2. The forward pointer of the current control block contains one of the 'End of Chain' values. These values are:
  - 00000000 for a 31-bit Start Address or 0000000000000000 for a 64-bit Start Address.
  - FFFFFFFF for a 31-bit Start Address or FFFFFFFFFFFFFFFF for a 64-bit Start Address.
  - The initial start address—implying the chain has looped

- A user supplied End Of Chain Identifier:
  - The value specified is automatically padded with leading zeroes to a length of 8 digits for a 31-bit Start Address or 16 digits for a 64-bit Start Address. A 64-bit address might include an underscore (see Start Address above for details).
  - If more than 8 digits are specified for a 31-bit Start Address, then the value is left truncated to 8 digits.
  - If more than 16 digits are specified for a 64-bit Start Address, and the address does not include an underscore, then the value is left truncated to 16 digits.
- 3. The forward pointer of the current control block points to invalid or unavailable storage.

For each control block, its address and the first 32 bytes of data are shown.

Optionally, you can provide an eyecatcher and its offset in the control block, in which case, for each control block, the text at the specified offset is compared against the supplied text, and if they do not match, then a warning message is issued.

As an example of the RUNCHAIN command, the Storage RUNCHAIN Command entry display might be specified as follows:

File View Services Help Storage RUNCHAIN Command
<p>Enter the required fields and press Enter .</p> <p>           Start Address . . . . . <u>0005C000</u>            Max Number Control Blocks <u>9999</u> (Decimal)            Forward Pointer Offset . . . <u>1C</u> (Hex)            End of Chain Identifier . . . _____ (Hex, Default Values: All 0's and all F's)            Eyecatcher Text . . . . . <u>DFHSMQPH</u>            Eyecatcher Offset . . . . . <u>2</u> (Hex)         </p> <p style="margin-top: 20px;">           F1=Help    F3=Exit    F12=Cancel         </p> <p style="margin-top: 10px;">           000011    n/a    01 PARM1    PIC X(4) .            000012    n/a    01 PARM2.         </p> <p style="margin-top: 10px;"> <b>Data Field Values:</b>            PARM1 = 0001            PARM2 = HEADING FOR IDIXSNAP            F1=Help    F3=Exit    F5=RptFind    F6=Actions    F7=Up    F8=Down            F10=Left    F11=Right         </p>

Pressing Enter, the following display is presented:

## Displaying chained data areas

```
File View Services Help
Runchain starting at address 0005C000. 1 Control blocks          Line 1 Col 1 80
Command ===>                                                 Scroll ===> CSR
CICS DUMP: SYSTEM=QXPM2C61 CODE=ASRA      ID=      MVS2      2003/06/25 13:47:55

Address   Hex (First 32 Bytes Of Data)
0005C000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 002AA000 0005
0005B000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005C000 0005
0005A000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005B000 0005
00059000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005A000 2E80
2E80CCCC  Invalid eyecatcher. Expected: >DFHSMQPH
                         Found : 24.....Q.
2E80CCCC  C3C1F2F4 2E80CC14 1ED82728 C0010100 00000000 00000600 00059000 002A
002AA000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 2E80CCCC 0005
End of RUNCHAIN 6 Control Blocks processed

*** Bottom of data.

F1=Help      F3=Exit      F5=RptFind    F6=Actions    F7=Up       F8=Down
F10=Left     F11=Right
```

To exit from the RUNCHAIN command, enter EXIT (PF3).

## Disassembling object code

Using the DISASM command (see “DISASM” on page 65), you can disassemble object code at a given address. The DISASM command can be invoked either by entering DISASM on any interactive report command line, or by assigning DISASM to a PF key. When invoked, you are shown a popup panel similar to the following:

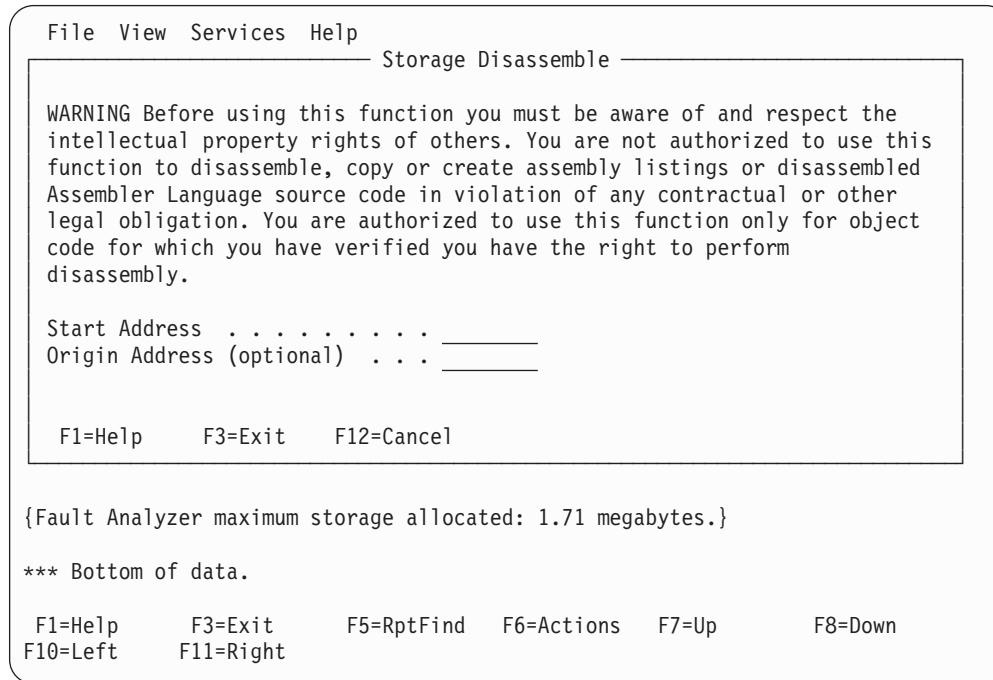
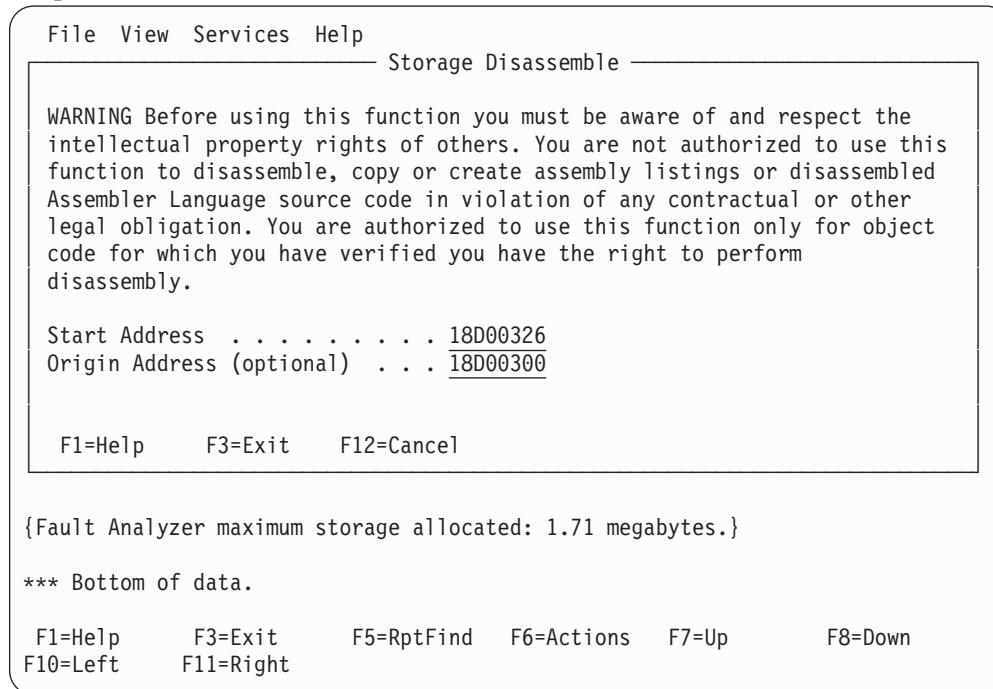


Figure 95. Sample Storage Disassemble display

The DISASM command attempts to disassemble code from a given start address. Optionally, an origin address can be provided in which case the offset of each disassembled instruction is calculated relative to the origin address, rather than the start address. If an origin address is not provided, then it defaults to the same as the start address.

As an example of the DISASM command, the Storage Disassemble display might be specified as follows:



Pressing Enter, the following display is presented:

## Disassembling object code

```
File View Services Help
Disassemble command Line 1 Col 1 80
Command ==> Scroll ==> CSR
JOBNAME: JA84Q13A SYSTEM ABEND: 0C7 IBM3 2003/12/01 13:59:00

Address Offset Hex Instruction
18D00326 +26 05EF BALR R14,R15
18D00328 +28 50B0 D0C8 ST R11,200(,R13)
18D0032C +2C 41A0 D0D4 LA R10,212(,R13)
18D00330 +30 D20F 4010 306C MVC 16(16,R4),108(R3)
18D00336 +36 D216 4020 3557 MVC 32(23,R4),1367(R3)
18D0033C +3C 58E0 303C L R14,60(,R3)
18D00340 +40 50E0 4038 ST R14,56(,R4)
18D00344 +44 5890 4038 L R9,56(,R4)
18D00348 +48 4090 403E STH R9,62(,R4)
18D0034C +4C 4190 4020 LA R9,32(,R4)
18D00350 +50 5090 4010 ST R9,16(,R4)
18D00354 +54 4170 D128 LA R7,296(,R13)
18D00358 +58 5070 4018 ST R7,24(,R4)
18D0035C +5C 4190 403E LA R9,62(,R4)

F1=Help F3=Exit F4=Dsect F5=RptFind F6=Actions F7=Up
F8=Down F10=Left F11=Right F12=retrieve
```

Once the panel showing the disassembled instructions has been displayed (see example above), then PF7 and PF8 can be used to scroll backwards and forwards.

To exit from the DISASM command, enter EXIT (PF3).

## Converting STORE CLOCK values

Using the STCK command (see “STCK” on page 74), you can convert binary STORE CLOCK values to human-readable date and time format. The STCK command can be invoked either by entering STCK on any interactive report command line, or by assigning STCK to a PF key. When invoked, you are shown a popup panel similar to the following:

## Converting STORE CLOCK values

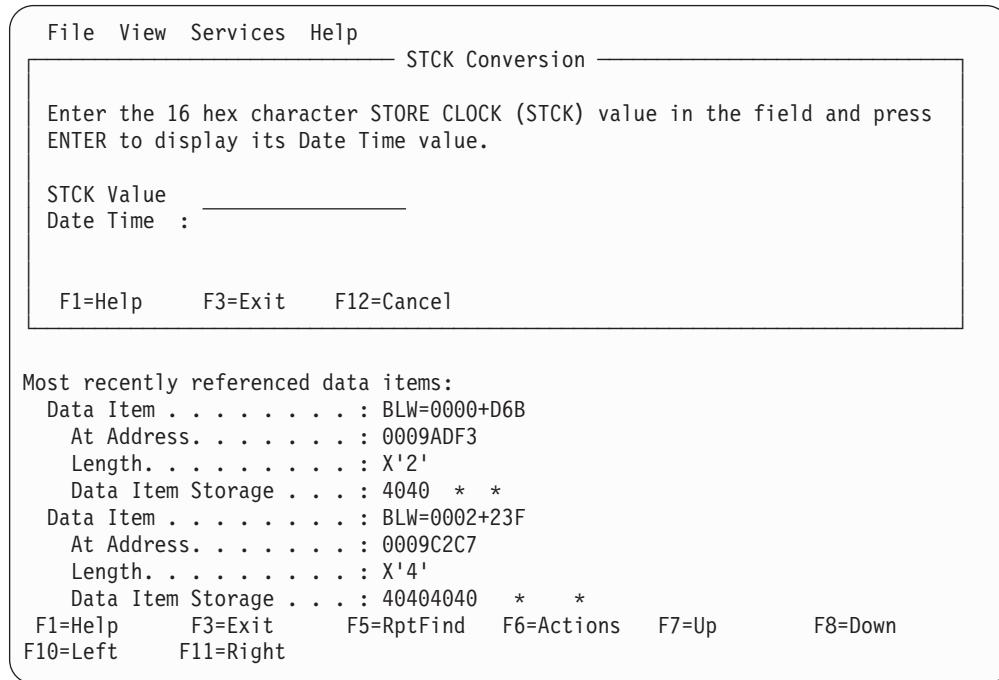
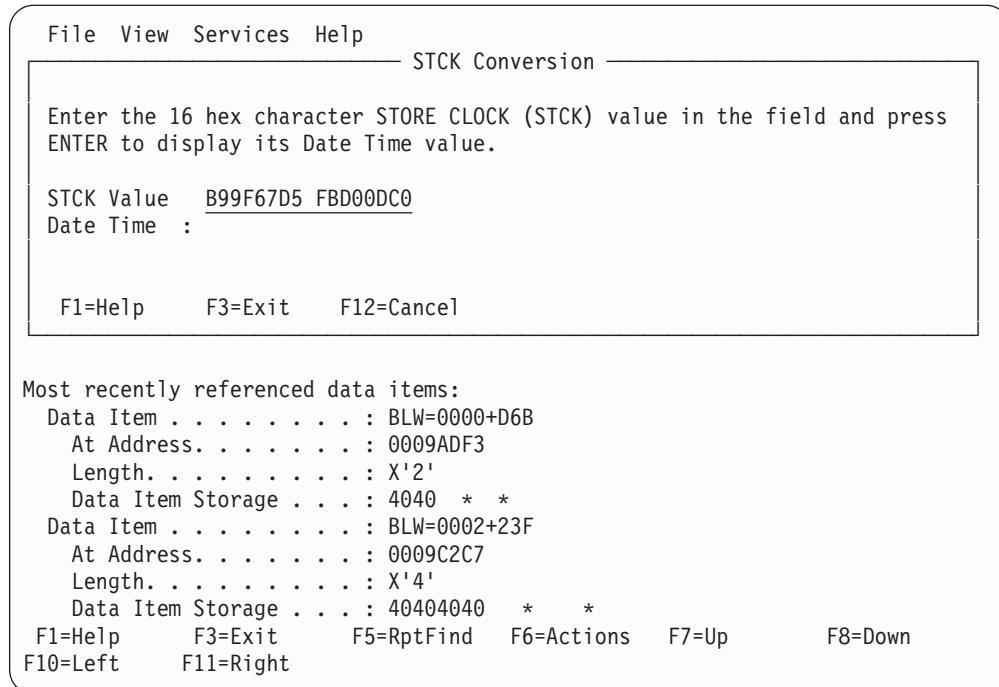


Figure 96. Sample STCK Conversion display

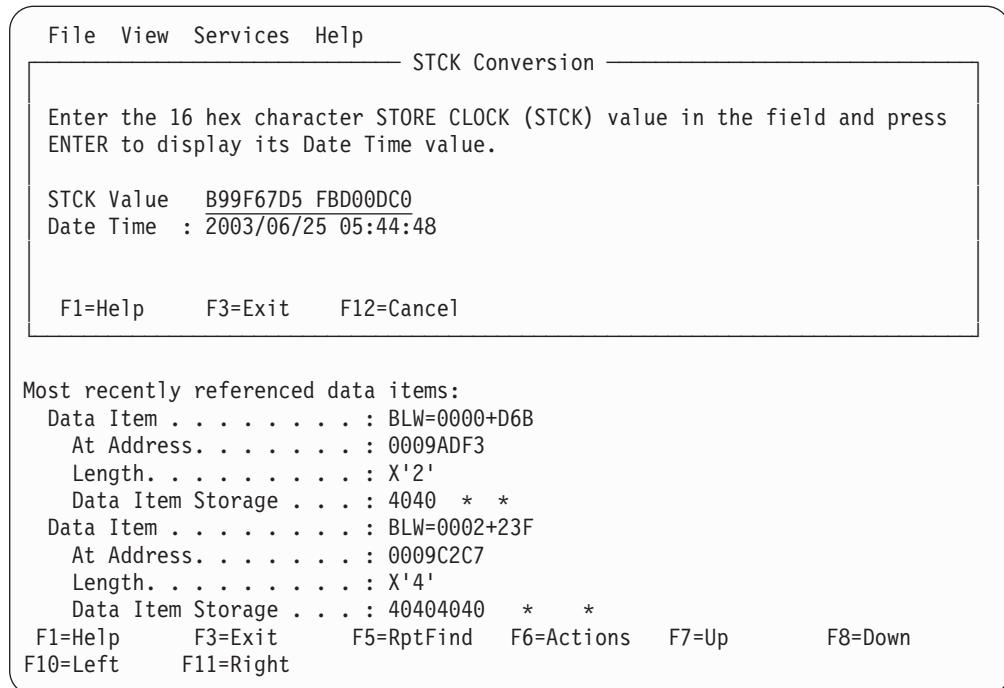
The STCK value must be entered as 16 hexadecimal characters. Any imbedded blanks are ignored.

As an example of the STCK command, the STCK Conversion display might be specified as follows:



Pressing Enter, the display is updated as follows:

## Converting STORE CLOCK values



To exit from the STCK command, enter EXIT (PF3).

## User-specific report formatting

Using the EXEC command (see “EXEC” on page 66), a REXX Formatting user exit can be executed. This type of exit is able to generate a display of user-specific information, such as formatting of data areas which are unique to the analyzed application environment. For general information about this type of exit, see “Formatting user exit” on page 381.

Non-REXX Formatting user exits cannot be executed through the EXEC command.

If an exit name is specified with the EXEC command, then that exit is executed and a display containing any information that the exit has provided is presented. However, if no exit name is specified, then a list of all REXX Formatting user exits which are available from the IDIEXEC concatenation of data sets is presented in a Formatting User Exit Selection List display like the following example:

## User-specific report formatting

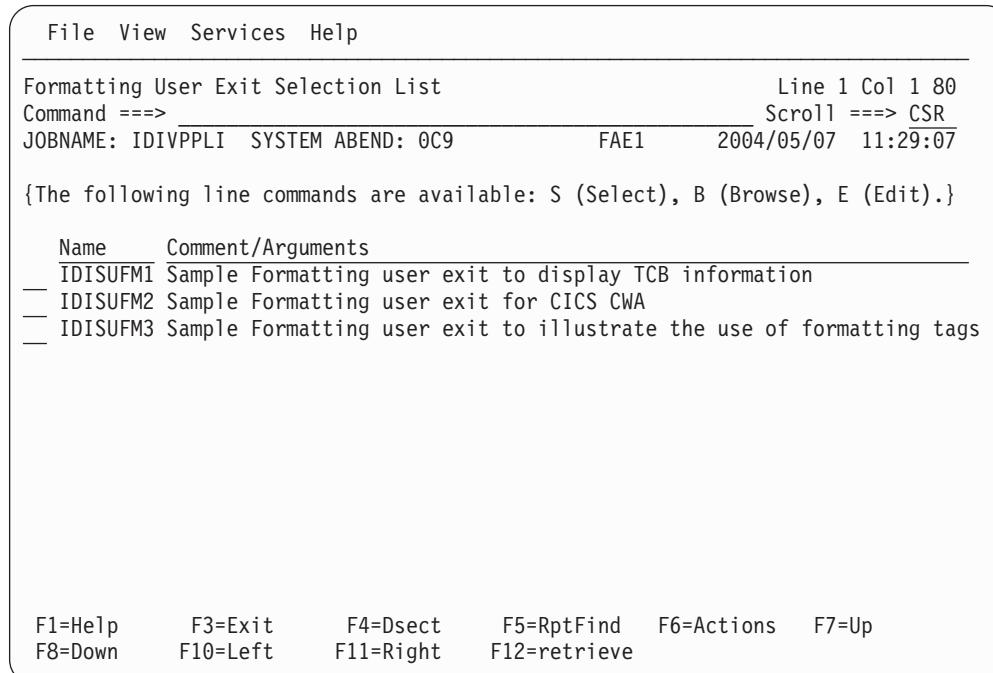


Figure 97. Sample Formatting User Exit Selection List display

To be recognized as a Formatting user exit for the Formatting User Exit Selection List display, the exit names must be specified in a control member within each IDIEXEC data set. The name of the control member must be \$\$UFMTX. Each record in the control member can contain the member name of the exit (not case-sensitive) and optionally a comment which is included in the Formatting User Exit Selection List display. A sample control member follows:

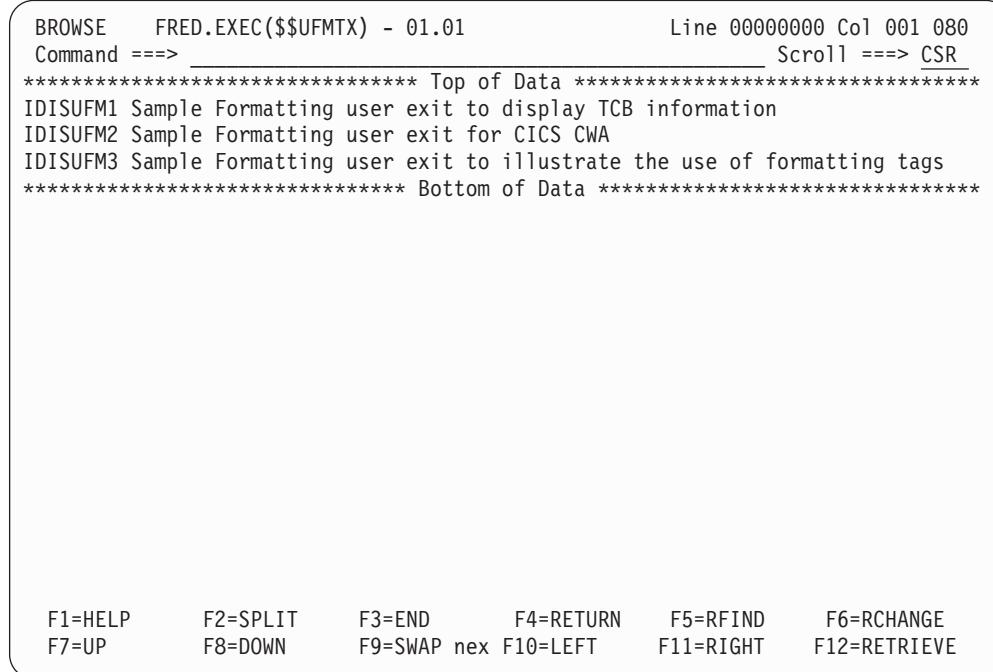


Figure 98. Sample \$\$UFMTX member

## User-specific report formatting

The above sample control member is provided in softcopy format as member IDISUFMX in data set IDLSIDISAM1.

To use this sample control member, you should:

1. Copy it to another data set and rename it to \$\$UFMTX.
2. Copy the three sample Formatting user exits named within it to the same data set.
3. Specify the data set name now containing the control member and the exits in the DATASETS(IDIEXEC(*data-set-name*)) option.
4. Invoke the Fault Analyzer ISPF interface and perform interactive reanalysis against a fault entry.
5. Issue the EXEC command without specifying an exit name. This action should present a display similar to that in Figure 97 on page 165 from where the sample exits can be run.

From the Formatting User Exit Selection List display, a line command can be issued against individual exits:

- S**      Executes the exit.  
**B**      Enters ISPF browse against the exit.  
**E**      Enters ISPF edit against the exit.

Often, exits require one or more parameters to be passed. Passing these parameters can be done by clearing (if necessary) and overtyping the "Comments/Arguments" field of the Formatting User Exit Selection List display to the right of the exit name. The field changes color when it has been overtyped to indicate that the data is used as parameters for the exit. By clearing the field, and pressing Enter, the original comment can be redisplayed instead.

---

## Prompting for compiler listing or side file

If no satisfactory compiler listing or side file was found for a COBOL, PL/I, C/C++, or assembler program, then a prompting display, as the example shown in Figure 99 on page 167, is presented. Prompting does not occur for C/C++ if the side file was originally located in HFS.

**Note:** In the following, the term "side file" refers to either a Problem Determination Tools LANGX side file, a COBOL SYSDEBUG side file that was generated by using the TEST(NONE,SYM,SEPARATE) compiler option, or an Enterprise PL/I side file that was generated by using the TEST(STMT,SYM,NOHOOK,SEPARATE) compiler option.

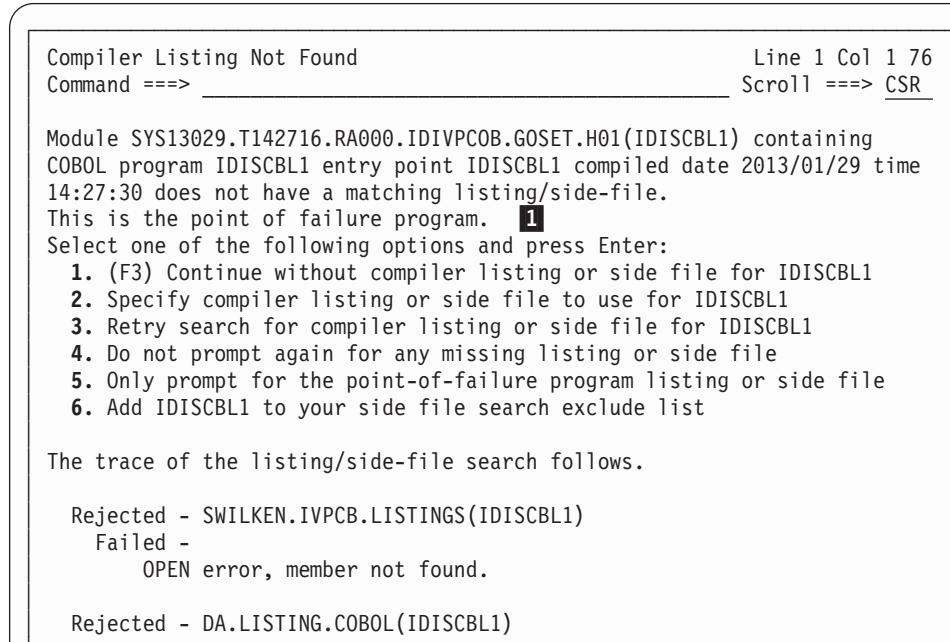


Figure 99. Sample Compiler Listing Not Found display

The prompt provides you with these choices:

**1. (F3) Continue without compiler listing or side file for *program-name***

If a compiler listing or side file cannot be supplied, select this option to continue without program source code information. Alternatively, enter the EXIT (PF3) or CANCEL (PF12) command.

**2. Specify compiler listing or side file to use for *program-name***

This option displays a pop-up panel in which you can provide the data set and member name (if a PDS(E) data set) of a compiler listing or side file (or, in the case of Enterprise COBOL version 5, a program object containing DWARF debugging information) that should be used for the current program as the example shown in Figure 100 on page 168.

## Prompting for compiler listing or side file

Specify Compiler Listing or Side File Command ==> _____	Line 1 Col 1 76 Scroll ==> CSR
<p>Specify the data set and member name containing the compiler listing or side file and press Enter.</p> <p>Data Set Name . . . : <b>'TDEV003.LISTING.PLI'</b> Member. . . . . : <b>IDISCB1</b></p> <p>Alternatively, place cursor on choice and press Enter to use previously specified compiler listing or side file data set name.</p> <p>==&gt; 'TDEV003.LISTING.PLI' ==&gt; 'TDEV003.0SDSF' ==&gt; 'PMR.P03527.B370.C000.IDILANG' ==&gt; 'TDEV003.JCLLIB' ==&gt; 'TDEV003.\$\$TEMP\$\$.LISTING' ==&gt; ==&gt; ==&gt; ==&gt;</p> <p>*** Bottom of data.</p>	

*Figure 100. Sample Specify Compiler Listing or Side File display*

The data set name is specified in accordance with the ISPF convention of prefixing with the current TSO prefix, unless enclosed in single quotes.

The last 10 data set names that were specified are stored in the ISPF profile for your application ID and are used for initialization of the display. The most recent data set name is used to initialize the data set name field, but any one of the listed data set names can be selected by placing the cursor on the desired data set name and pressing Enter.

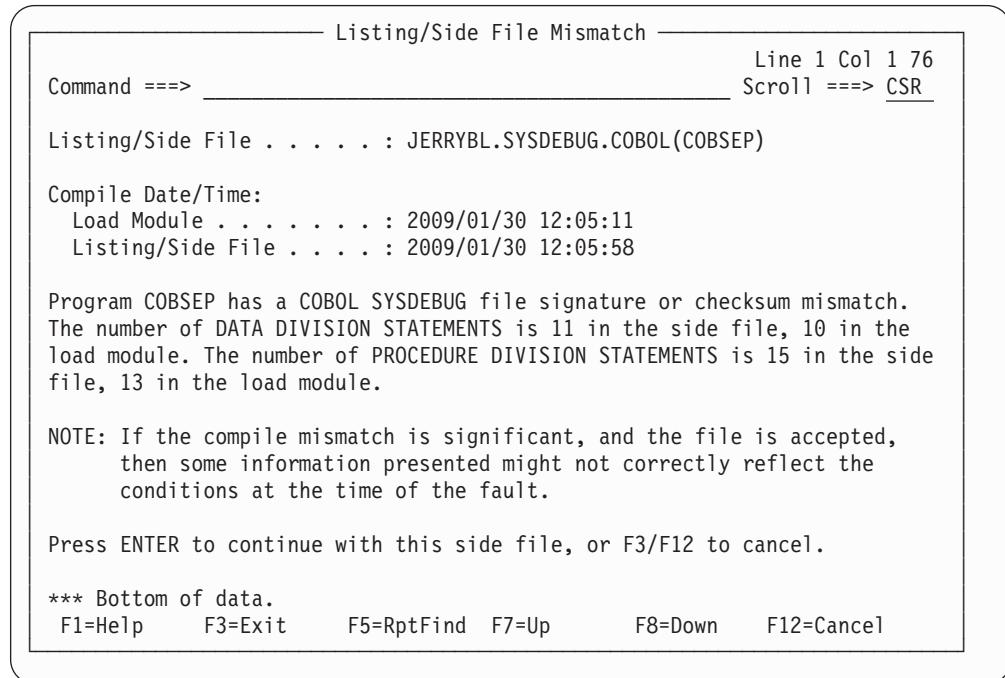
The member name defaults to the program name for which the listing or side file is required. If the actual member name for your listing or side file differs from the program name, you need to change this field.

If a sequential data set is specified, then the member name is ignored.

Having specified or selected the desired data set and member name, press the Enter key.

If Fault Analyzer has determined that the specified compiler listing or side file is not a good match, then another prompt as the example shown in Figure 101 on page 169 is presented.

## Prompting for compiler listing or side file



*Figure 101. Sample Listing/Side File Mismatch display*

If pressing Enter, and thus accepting the provided mismatching compiler listing or side file, then it is possible that some incorrect information is presented, for example, data fields with incorrect values, or incorrect source lines or statements.

In case a side file fails validation to such a degree that it is not even possible to attempt using it, then an Enter response to the prompt in Figure 101 instead shows the display in Figure 99 on page 167 again, which allows for a different side file to be provided.

3. Retry search for compiler listing or side file for *program-name*

Selecting this option causes Fault Analyzer to repeat the search for the compiler listing or side file via the standard search path. This option can be selected after, for example, having recompiled the current program via a split screen ISPF session and provided the compiler listing or side file to Fault Analyzer in, for example, the IDILCOB data set concatenation.

This repeated search is only performed once. The user is not prompted a second time for the same program, even if the listing or side file is still not found.

4. Do not prompt again for any missing listing or side file

If you select this option, then Fault Analyzer does not prompt you again for a missing compiler listing or side file for any program for the duration of the current interactive reanalysis session.

## 5. Only prompt for the point-of-failure program listing or side file

If you select this option, then Fault Analyzer only prompts you again for a missing compiler listing or side file for a program, if that program has been determined as belonging to the point-of-failure event. If the initial prompt is already for the point-of-failure program, then a message is added to the display to indicate this ( **1** ).

## 6. Add *program-name* to your side file search exclude list

If you select this option, then a display that enables you to add the current

## Prompting for compiler listing or side file

program name to your side file search exclude list, as the example shown in Figure 102, is presented.

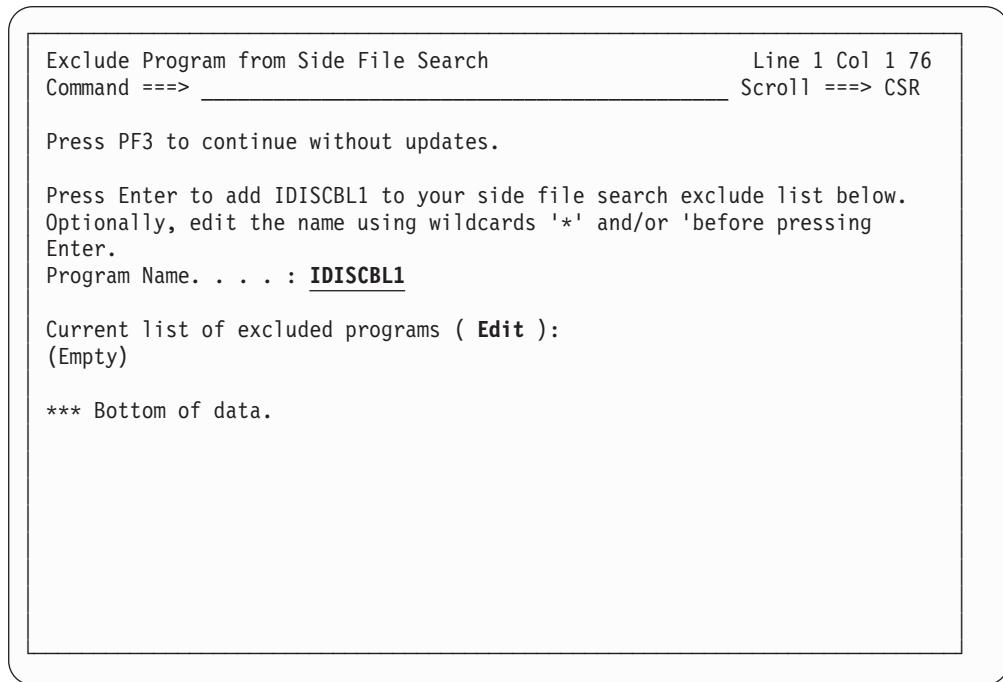


Figure 102. Sample Exclude Program from Side File Search display

Press PF3 to continue without updates. You might be prompted again to provide a compiler listing or side file for this program.

Press Enter to add the program name to your side file search exclude list. Optionally, edit the name first using the wildcard characters '\*' (zero, one or more characters) or '%' (a single required character) to make the name more generic.

Your current side file search exclude list is provided. The list can be modified by placing the cursor on the Edit point-and-shoot field and pressing Enter. The program names must be separated by one or more blanks, and can include the wildcard characters '\*' (zero, one or more characters) and '%' (a single required character). The specified program names are not case sensitive.

The following are examples of valid program name specifications:

\*XMAI\*  
PAYROLL0  
SELOPT%  
SUBRTN\*

The program name exclude list can also be edited from the Interactive Reanalysis Options display—for details, see “Interactive reanalysis options” on page 103.

Following the list of options is the trace of the listing/side file search.

This trace is equivalent to what can be obtained when using the IDITRACE DDname, as shown in “IDITRACE information” on page 292.

When the Compiler Listing Not Found display (Figure 99 on page 167) is first shown, then the entire search trace up until that point in time is provided. From then on, the trace only contains the records that were written since the last time when the Compiler Listing Not Found display was shown.

### Controlling prompting

The interactive reanalysis option, "Prompt for missing side files", determines if prompting occurs or not. For details about this option, see "Interactive reanalysis options" on page 103.

---

## Data sets used for interactive reanalysis

When performing interactive reanalysis through the ISPF interface, pre-allocation is performed as required for any Fault Analyzer compiler listing or side file data sets that were used in real time. Allocations are performed for Fault Analyzer data sets if they were explicitly included in the real-time JCL, or supplied through the DataSets option or an Analysis Control user exit. These data sets are used in the reanalysis in an attempt to recreate the same execution environment as were used in real-time.

DataSets options that are specified via the IDIOPTS user options file or the PARM field cause those data sets to be logically concatenated to the data sets from the real-time execution.

If the "Display panel to alter allocated data sets" option on the "Interactive options" display is set to Y (see "Interactive reanalysis options" on page 103), then it is possible to make changes to the real-time data set specifications before initiating the reanalysis. Also, any data sets that were used in real time but do not exist in the reanalysis environment, or data sets with READ access prohibited, are identified by a comment as shown in the following example for IDILCOB:

```
/* The following IDILCOB data set is unavailable:  
/*      DD DISP=SHR,DSN=D01.COBOL.LISTINGS  
/* The following IDILCOB data set is READ protected:  
/*      DD DISP=SHR,DSN=CTEST.PROTECT.LISTINGS
```

---

## Refresh processing

Refresh processing is what occurs when a fault entry is being rewritten due to user information, having been updated during interactive reanalysis. The user information which might cause this refresh processing is any of the following:

- User name, user title, or lock flag changed via the INFO command or the "File->Fault Entry Information" action-bar pull-down menu option.
- User notes against dump storage addresses added, deleted, or modified.

When user information has changed, then a display like the example shown in Figure 103 on page 172 is presented upon exiting the interactive report, which permits the cancellation of the refresh, or the suppression of the minidump.

## Refresh processing

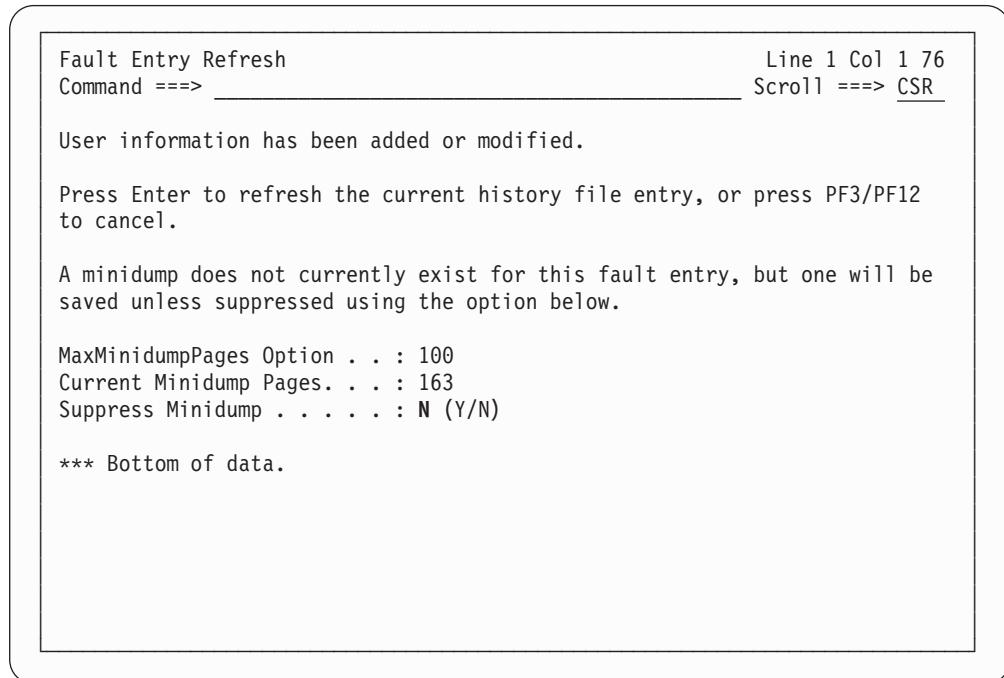


Figure 103. Sample refresh processing exit prompt

The option to suppress the minidump is only included if the fault does not yet include a minidump and the current minidump size exceeds the MaxMinidumpPages option limit in effect.

If instead the reanalysis is performed in batch, then a user exit can be used to perform the equivalent function. This exit is effectively an End Processing user exit, and is specified using the RefreshExits option (see “RefreshExits” on page 496).

## COBOL Explorer

COBOL Explorer uses the event source line to create a branch analysis that shows a procedure traceback and use of selected variables. References that modify selected variables are highlighted. This set includes modify by group or REDEFINES.

The traditional editor-based source program presentation is replaced by a collapsed or expanded source view where you see just the parts of the program that are relevant. This approach makes working with large COBOL programs a snap.

There are three ways to start COBOL Explorer:

- From the point-and-shoot field that is in the Event Details display for a COBOL program event.
- From the Services pull-down menu.
- By entering the CE command anywhere during interactive reanalysis.

If you enter the CE command without a parameter, or start COBOL Explorer from the Services pull-down menu, a pop-up list is displayed. Select a program from the pop-up.

## COBOL Explorer example

The following is an example of how to use the COBOL Explorer feature of Fault Analyzer.

A fault entry for an abend in a COBOL program is analyzed interactively using the 'T' line command from the Fault Entry List display. The "Event Summary" is selected, and from there the COBOL program event is selected, resulting in the "Event Details" display shown in Figure 104.

```

File View Services Help
Event 1 of 1: Abend S0CB *** Point of Failure ***
Command ==> _____ Line 1 Col 1 80
JOBNAME: COBPERF6 SYSTEM ABEND: 0CF FAE1 2009/07/21 16 Scroll ==> CSR

Abend Code. . . . . : S0CB
Program-Interruption Code . : 000B (Decimal-Divide Exception)
The divisor was zero in a signed decimal division.

The source code below was executed via the following sequence of PERFORM
statements:
Source
Line #
000105      PERFORM READ-FILE UNTIL END-OF-FILE = '1'.
000122      NOT AT END PERFORM PROCESS-VEHICLE
000133      PERFORM PROCESS-CAR
000151      PERFORM CALC-TAX.

COBOL Source Code:
Source
Line #
000175      COMPUTE BASE-AMOUNT = PRICE / CC

Data Field Declarations:
Source
Line #
000033      03 PRICE          PIC 9(6).
000049      07 CC            PIC 9(4).
000094      01 BASE-AMOUNT    PIC 9(3)V99 COMP-3.

Data Field Values:
BASE-AMOUNT = 0.00
CC          = 0   *** Cause of error ***
PRICE       = 50000

COBOL Explorer 1

The listing file used for the above was found in
MY.LISTING.COBOL(COBPERF6).

:

```

Figure 104. Sample COBOL Explorer Event Details display

By selecting the "COBOL Explorer" point-and-shoot field shown at **1**, the "Source Line and Data Values" display in Figure 105 on page 174 is shown.

## COBOL Explorer

```
Source Line and Data Values                                Line 1 Col 1 76
Command ===> _____                                     Scroll ===> CSR

Source:
000175          COMPUTE BASE-AMOUNT = PRICE / CC

Data Field Declarations:
000033          03 PRICE             PIC 9(6).
000049          07 CC               PIC 9(4).
000094          01 BASE-AMOUNT      PIC 9(3)V99 COMP-3.

Data Field Values:
s BASE-AMOUNT = 0.00
s CC           = 0 *** Cause of error ***  2
s PRICE        = 50000

Select data fields to use with source line to create view.

*** Bottom of data.
```

Figure 105. Sample COBOL Explorer Source Line and Data Values display

Because the data field "CC" at **2** has been identified as the cause of the error (divide by zero), we want to look at this data field more closely. To do this, unselect the other two data fields ("BASE-AMOUNT" and "PRICE") by over-typing with blanks, and press Enter. The result is the display shown in Figure 106.

```
File View Services Help

Exploring COBOL Program COBEX1                               Line 1 Col 1 80
Command ===> _____                                     Scroll ===> CSR
JOBNAME: COBEX1   SYSTEM ABEND: 0CB          FAE1   2014/04/14 20:40:31
+ Expand all / - Collapse all   - Comments + Level 88 - Redefines

+ 000001 IDENTIFICATION DIVISION.
+ 000003 ENVIRONMENT DIVISION.
+ 000011 FILE SECTION.
+ 000017 WORKING-STORAGE SECTION.
+ 000026 01 VEHICLE-RECORD.
000038          03 SPECIFICATION.
000042          05 ENGINE.
000049          07 CC             PIC 9(4).
+ 000081 01 HEADINGS REDEFINES VEHICLE-RECORD.
+ 000084 01 WS-REC REDEFINES VEHICLE-RECORD.
+ 000098 LINKAGE SECTION.
+ 000101 PROCEDURE DIVISION USING PARMs.
000105          PERFORM READ-FILE UNTIL END-OF-FILE = '1'.
+ 000119 READ-FILE.
000122          NOT AT END PERFORM PROCESS-VEHICLE
+ 000125 PROCESS-VEHICLE.
000126      MOVE FS-REC TO WS-REC.
000133          PERFORM PROCESS-CAR
+ 000149 PROCESS-CAR.
000151          PERFORM CALC-TAX.
+ 000173 CALC-TAX.
000175      COMPUTE BASE-AMOUNT = PRICE / CC
000178          COMPUTE BASE-AMOUNT = CC / CYLINDERS
```

Figure 106. Sample COBOL Explorer Debug View display

The above Debug View display consists of a branch-analysis showing how the program got to the source line, and any source lines that reference the selected variable (or variables, if more than one had been selected from the previous Source Line and Data Values display).

This view shows that the abending source line 175 is in procedure CALC\_TAX, and the branch analysis reveals the execution path to the source line.

In addition to selected variables, any references to containing group items or redefines are also shown. Any source lines that modify a reference are highlighted (often these are used to create a new view as variables that modify referenced variables might be of particular interest, use PF10/PF11 to scroll views).

You can select the following by placing the cursor on the appropriate line and pressing Enter (or just double-click if you have set pComm->Edit->Preferences->Hotspots->Fnn + Point and select):

- Select any highlighted source line to create a new view.
- Select a DATA DIVISION section (for example, WORKING-STORAGE SECTION) to invoke the Associated Storage Areas display for that section (this shows all variables and their values). The FILE SECTION also allows you to edit/browse/view open files with ISPF or File Manager.
- Select a data item to show all references to it and the section/paragraph in which the reference occurs. For group items and group members, an aggregate map is also shown.
- Select a procedure name to show all branches to it and the procedure in which the branch occurs.
- Select an executable source line to show the value of its variables and optionally create a new view for that source line and selected variables. If the source line is a branch (for example, PERFORM, GO TO), the target procedure is expanded (subsequent collapse returns to the branch statement).
- Select an MQI call (MQOPEN/MQGET/MQPUT/MQCLOSE) to edit/browse the queue or list queue managers with File Manager WebSphere MQ.

Section Expand/Collapse, by means of placing the cursor on the '+' or '-' signs on the left of the display and pressing Enter, is used to explore the program. Sections comprise FILE SECTION, WORKING-STORAGE SECTION and so on, plus any sections and paragraphs in the PROCEDURE DIVISION.

DATA DIVISION level 01 group items are initially collapsed. These can be expanded to show group members.

Expand the PROCEDURE DIVISION to reveal all procedures. These can then be expanded individually to show source.

Global Expand/Collapse, by means of placing the cursor on the '+' or '-' signs at the top of the display and pressing Enter, provides more filtering of the source program:

- "Expand all" expands all sections and group items. (Note that this causes all source records to be searched when using the Find command.)
- "Collapse all" restores the view's initial display.
- "Comments" expands or collapses comment lines.
- "Level 88" expands or collapses level 88 items showing only the high-level item when collapsed.

## **COBOL Explorer**

- "Redefines" expands or collapses REDEFINES showing only the high-level item when collapsed.

View the video introducing COBOL Explorer at YouTube: [COBOL Explorer](#).

## **Deferred Breakpoints Feature**

COBOL Explorer supports the setting of Debug Tool Deferred Breakpoints, similar to IPVLANGP. For details, see "Common Component Customization Guide and User Guide".

To set Deferred Breakpoints, PD Tools Common Component must be installed.

---

## Chapter 6. Performing CICS system abend dump analysis

A feature unique to the interactive component of Fault Analyzer is the ability to analyze information that is related to CICS system abends. This chapter tells you how to:

- Set options for CICS system abend analysis
- Select a CICS system abend dump data set for analysis
- Display the resulting CICS system abend interactive report
- Create a history file entry for the analyzed dump data set

**Note:** The TSO region size required to analyze a CICS system dump is usually significantly greater than that required to reanalyze a normal fault entry.

The steps outlined in the following assume that you have already started the interactive component of Fault Analyzer from an ISPF session.

---

### Setting options for CICS system abend analysis

The general interactive reanalysis options are also used for CICS system abend analysis (see “Interactive reanalysis options” on page 103).

#### User exits

Since CICS system abend analysis is performed as reanalysis against an MVS SVC dump or SYSMDUMP data set, only the following user exits can be used:

- “Analysis Control user exit” on page 368
- “Compiler Listing Read user exit” on page 372
- “Message and Abend Code Explanation user exit” on page 377
- “Formatting user exit” on page 381

---

### Selecting a CICS dump data set

To select a CICS system abend SVC dump or SYSMDUMP data set, first select the Analyze MVS Dump Data Set option from the Fault Entry List display File menu (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This option brings up the Analyze MVS Dump Data Set display as shown in Figure 107 on page 178.

## Selecting a CICS dump data set

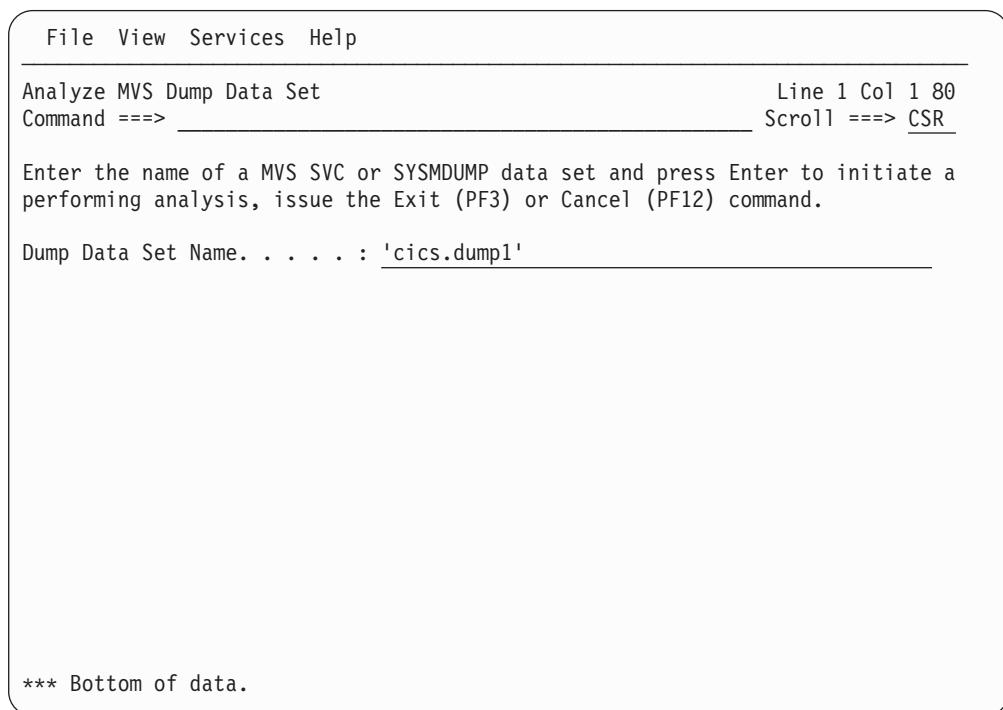


Figure 107. Sample Analyze MVS Dump Data Set display

On this display, type the name of the SVC dump or SYSMDUMP data set containing the CICS system abend dump to be analyzed. The data set name-specification follows the ISPF data set name rules, that is, a data set name that is not enclosed in single quotes is prefixed by the current TSO profile prefix. The data set specified is checked for existence before being accepted. In this example, data set CICS.DUMP1 is being analyzed.

The last data set name that is specified is stored in the ISPF profile for your application ID and is used for initialization of the display.

When the specified MVS dump data set name has been validated, the Fault Analyzer CICS system abend analysis commences as indicated by the "Analyzing MVS dump data set. Please wait..." message being displayed (Figure 108 on page 179).

## Selecting an address space to analyze

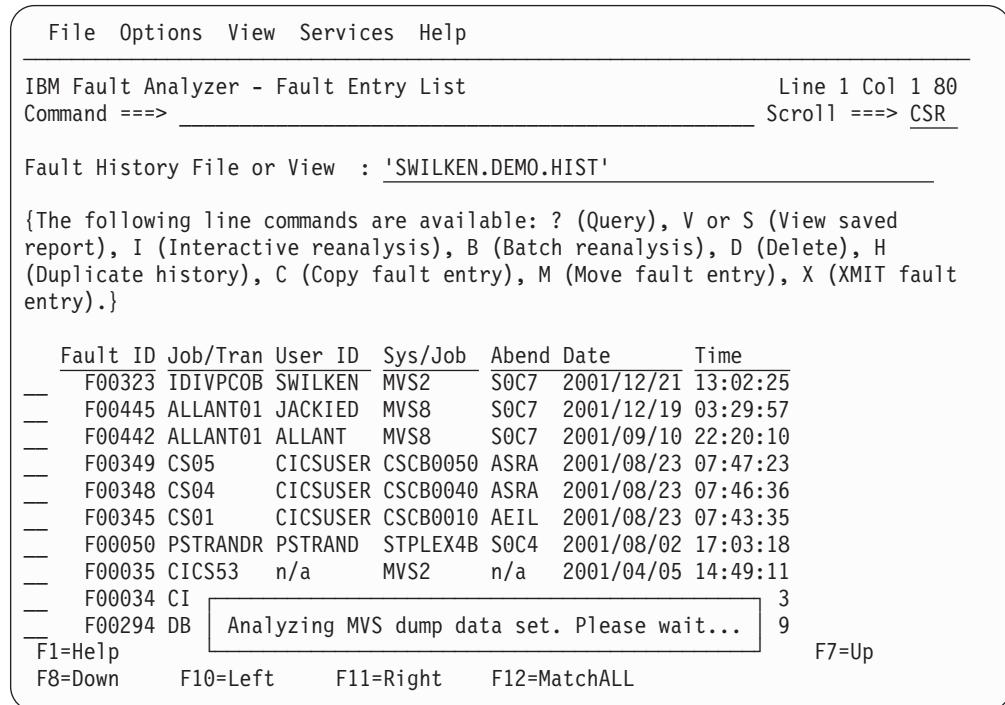


Figure 108. Recognizing that analysis has commenced

## Selecting an address space to analyze

If analyzing a dump containing multiple address spaces, then a selection list is displayed, as the example shown in Figure 109, from which a selection can be made by typing an S against the desired address space, and pressing Enter.

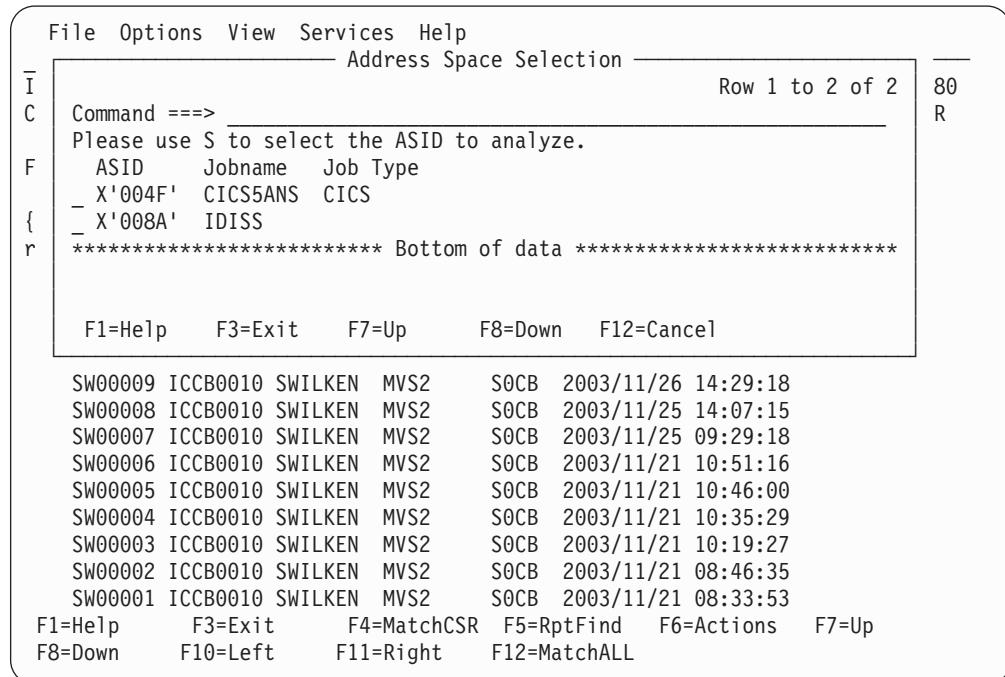


Figure 109. CICS system dump analysis address space selection

Any address spaces in which CICS was found to be active are marked with a job type of "CICS".

## Selecting an address space to analyze

If the dump contains only one address space, then no address space selection display is presented.

## Displaying the CICS system abend interactive report

When the analysis of the CICS system abend has completed, the CICS system abend interactive report is shown (Figure 110).

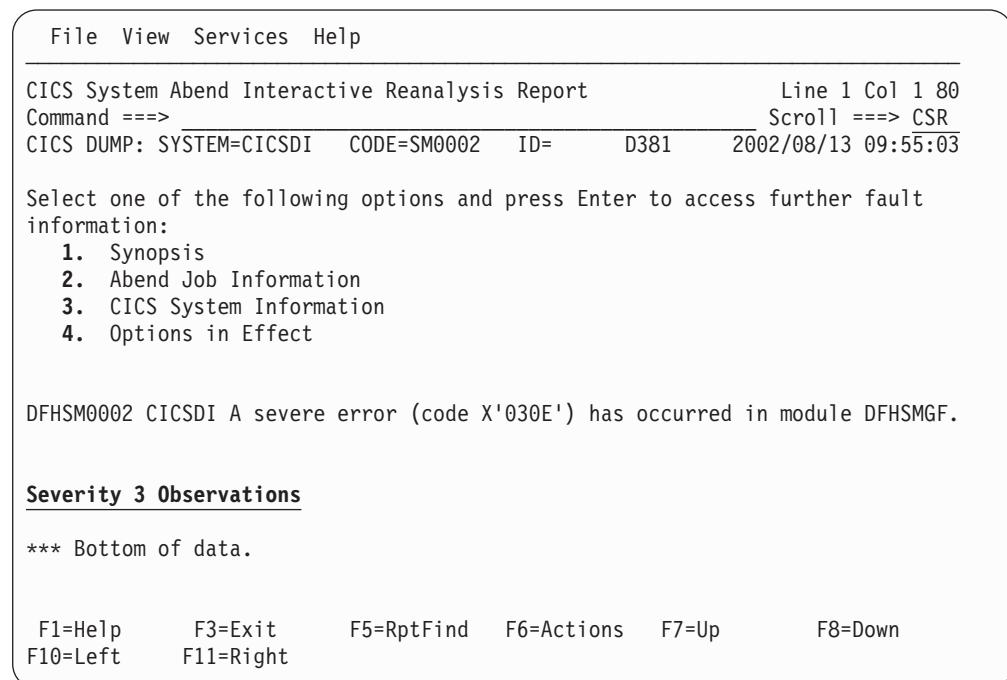


Figure 110. Sample CICS System Abend Interactive Reanalysis Report display

The initial interactive report display for a CICS system abend is different from that of any other fault type. The content is logically divided into the following sections:

### Available options

At the top of the display are options that can be selected by placing the cursor on the option number and pressing the Enter key. The tab key can be used to position the cursor to the option that you want. The individual options are explained in the sections that follow.

### Dump reason

If available, the list of options is followed by the reason why the analyzed dump was taken.

### Analysis observations

If any, the dump reason is followed by observations made during the analysis. These are loosely divided into three categories based on severity:

#### Severity

#### Description

- 1 Problems that are likely to be reasons for the fault.
- 2 Problems that might be reasons for the fault.

If ten or more severity 2 observations are available, then a point-and-shoot link that takes you to a separate display is provided instead.

## Displaying the CICS system abend interactive report

- 3 Problems that generally would not be considered contributing reasons for the fault, but merely items of interest.

If five or more severity 3 observations are available, then a point-and-shoot link that takes you to a separate display is provided instead.

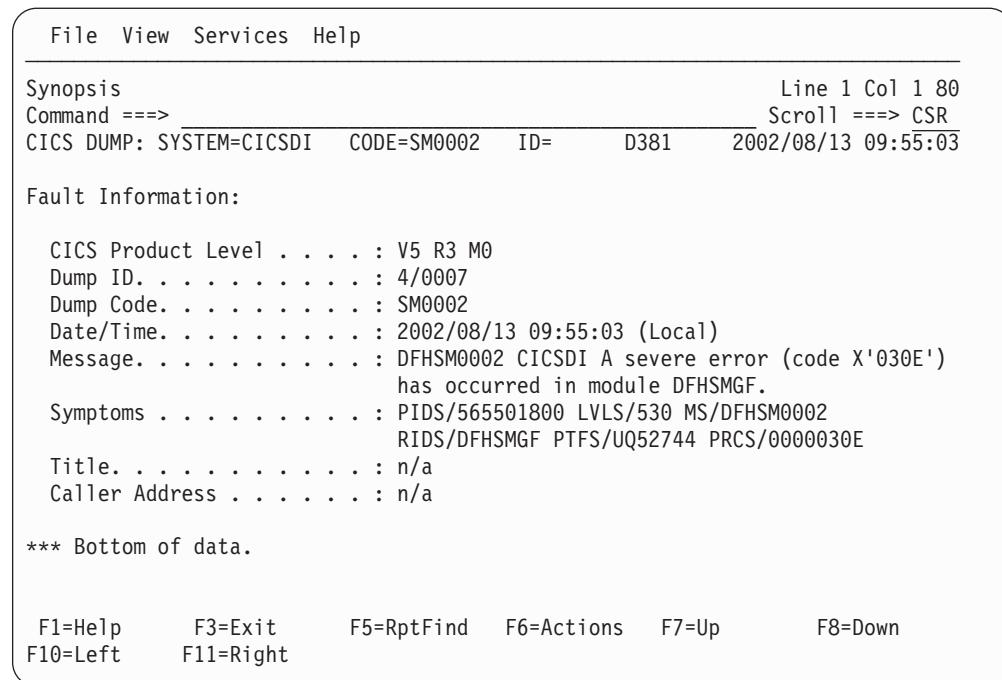
# Fastpath navigation

To make navigation easier in the CICS system dump analysis displays, fastpath commands can be entered on any command line. The valid fastpath commands are all of the highlighted fields shown in the CICS System Information display (an example is shown in Figure 113 on page 183).

If a fastpath command is prefixed by an exclamation mark (!), then it is the equivalent to going back to the CICS System Abend Interactive Reanalysis Report display and then entering the fastpath command. That is, it is analogous to using the jump command (=) in ISPF.

## **Option 1: Synopsis**

Selecting option 1 from the CICS System Abend Interactive Reanalysis Report display results in the presentation of the CICS System Abend Synopsis display, of which an example is shown in Figure 111.



*Figure 111. Sample CICS System Abend Synopsis display*

This display provides details about the analyzed dump.

## **Option 2: Abend Job Information**

Selecting option 2 from the CICS System Abend Interactive Reanalysis Report display results in the presentation of the Abend Job Information display, of which an example is shown in Figure 112 on page 182.

## Displaying the CICS system abend interactive report

The screenshot shows a window titled "File View Services Help". Below the menu bar, it displays "Abend Job Information" and "Line 1 Col 1 80". It also shows "Command ==> CICS DUMP: SYSTEM=CICSDI CODE=SM0002 ID= D381 2002/08/13 09:55:03" and "Scroll ==> CSR". The main content area is titled "IBM Fault Analyzer Abend Job information:" and lists the following parameters:

Abend Date . . . . .	:	2002/08/13
Abend Time . . . . .	:	09:55:03
System Name. . . . .	:	D381
Subsystem Info . . . . .	:	CICS V5 R3 M0
Job Name . . . . .	:	CICSDI
Job Step Name. . . . .	:	CICSDI
Exec Program Name. . . . .	:	DFHSIP
User ID. . . . .	:	ATICICS

Below this, another section titled "Execution Environment:" lists:

Operating System . . . . .	:	OS/390 V02R10M00
Data Facility Product. . . . .	:	DFSMS OS/390 V2R10M0
CPU Model. . . . .	:	2064

At the bottom of the display, there are function key definitions:

F1=Help	F3=Exit	F5=RptFind	F6=Actions	F7=Up	F8=Down
F10=Left	F11=Right				

Figure 112. Sample Abend Job Information display

This display provides information about the environment that existed when the abend occurred.

## Option 3: CICS System Information

Selecting option 3 from the CICS System Abend Interactive Reanalysis Report display results in the presentation of the CICS System Information display, of which an example is shown in Figure 113 on page 183.

## Displaying the CICS system abend interactive report

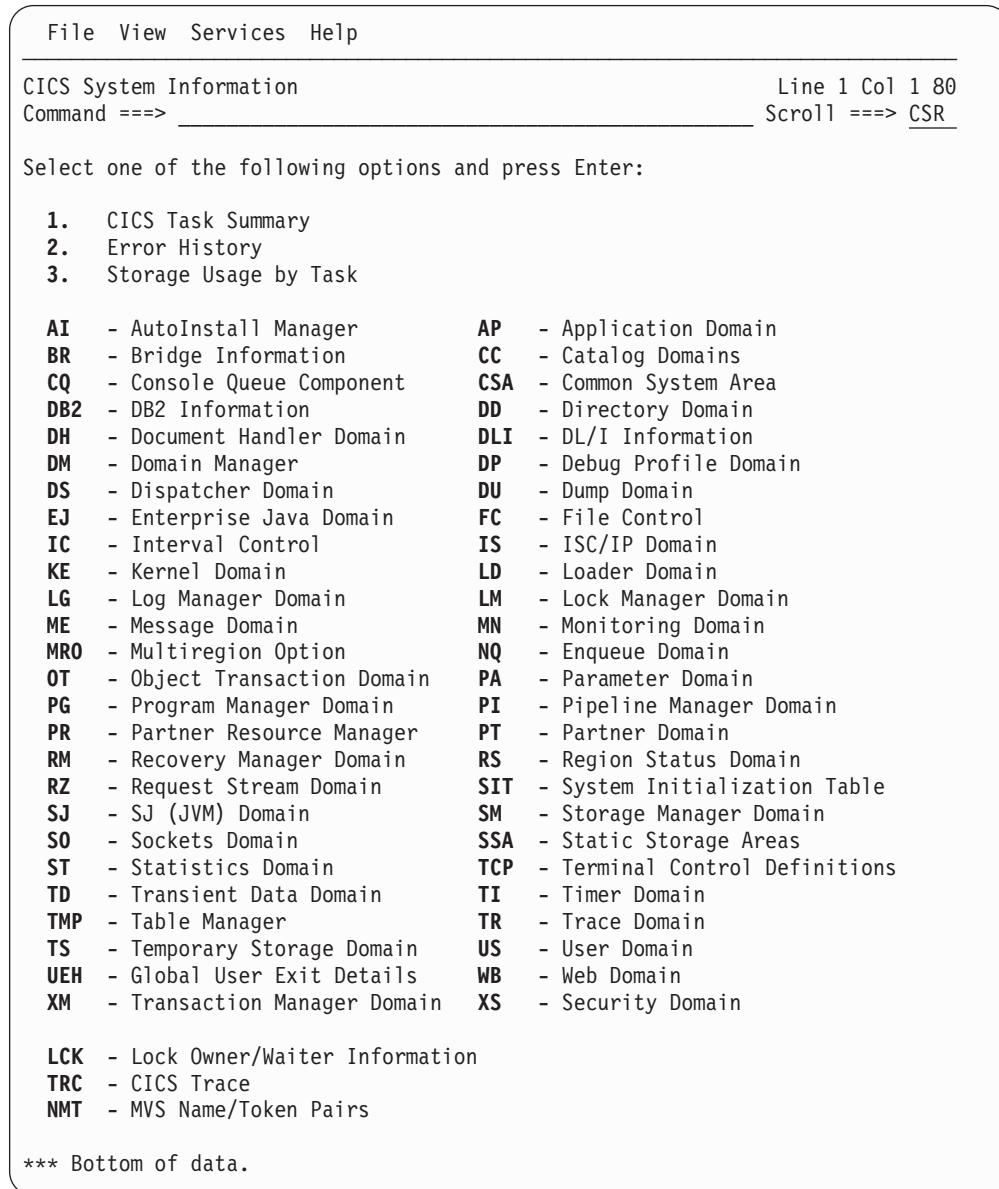


Figure 113. Sample CICS System Information display

This display provides options to select CICS system information of interest.

### Sorting and matching table displays

Whenever table headings are tabbable and shown in reverse highlight mode, then this highlighting indicates that the table column attributes are modifiable. By placing the cursor on the heading, and pressing Enter, a Column Attributes display is presented. This display allows you to sort the column data in ascending or descending order, or to show only the table rows that satisfy a given MATCH criteria.

The MATCH attribute is case insensitive and permits the use of wildcards. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character. Only table data rows that match the specified string are shown.

## Displaying the CICS system abend interactive report

All attribute settings are cumulative, which means that once a particular sort order is applied to a column, a sort on a different column is performed against the already sorted table data. Also, once table data is removed from the display due to a non-matching match criteria, the only way to restore this data is to perform a reset.

The reset can be performed by placing the cursor on the reset point-and-shoot field and pressing the Enter key. Alternatively, a RESET command can be entered on the command line to reset all tables in a given display simultaneously.

Columns for which attributes have been changed are shown with turquoise color instead of blue.

The most recent attribute setting is shown whenever a column header is selected.

## Option 4: Options in Effect

Selecting option 4 from the CICS System Abend Interactive Reanalysis Report display results in the presentation of the Options in Effect display, of which an example is shown in Figure 114 on page 185.

## Displaying the CICS system abend interactive report

```
File View Services Help
Options in Effect Line 1 Col 1 80
Command ==> Scroll ==> CSR
CICS DUMP: SYSTEM=CICSDI CODE=SM0002 ID= D381 2002/08/13 09:55:03

IBM Fault Analyzer Options in Effect:

{These are the options that were used to generate the current interactive
reanalysis report. To change any options, first return to the Fault Entry
List display and select "Interactive Reanalysis Options" from the "Options"
action-bar pull-down menu; then perform interactive reanalysis again.}

DumpDSN(CICS.DUMP1)
NoErrorHandler
Language(ENU)
NoLocale
NoPermitLangx

Data Sets:

{The following Fault Analyzer data set or path names were either
preallocated, specified via DataSets options, or provided as defaults.}

DDname Data Set or Path Name
IDIADATA PROD.SYSADATA
IDIDOC IDI.SIDIDOC1
IDIDSECT IDI.DSECTS
IDIEEXEC IBMUSER.EXEC
IDIHIST IDI.HIST
IDILC PROD.LISTING.C
IDILCOB PROD.LISTING.COOL
IDILPLI PROD.LISTING.PLI
IDIMAPS IDI.SIDIMAPS
IDIVSENU IDI.IDIHVENU

Exits:

{The following user exits were specified via Exits options.}

Type Name Type Invoked
NOTIFY NOTIFY REXX (Not applicable)
```

Figure 114. Sample Options in Effect display

This display provides information about Fault Analyzer options in effect for this reanalysis report.

## Creating a history file entry

When you exit from the CICS System Abend Interactive Reanalysis Report, you are presented with the option to create a history file entry as shown in Figure 115 on page 186.

## Creating a history file entry

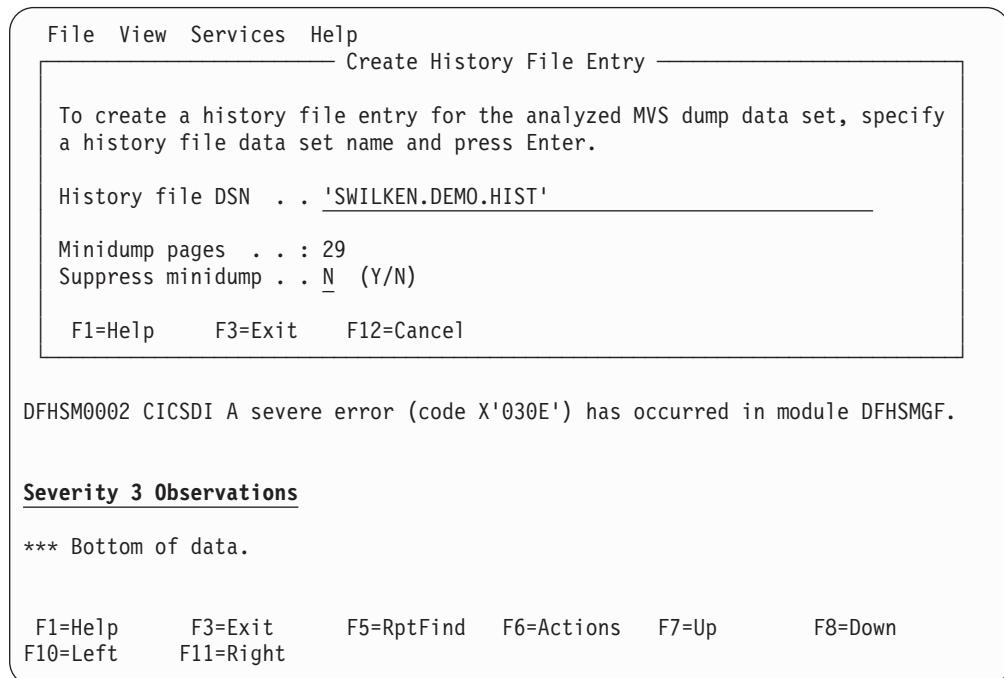


Figure 115. Sample Create History File Entry display

To create a history file entry:

1. If necessary, change the supplied name of the history file (obey the ISPF data set name specification rules).
2. Optionally change the "Suppress Minidump" option to 'Y' to show that a minidump with the indicated number of pages must not be created.
3. Press Enter.

The "history file DSN" field is by default initialized with the current history file name that is selected on the Fault Entry List display. If a view is being used, then the first history file name that is specified in the view member is used. To avoid creating a history file entry, press either F3 or F12.

If a history file entry was created, message IDI0003I is issued to inform you of the assigned fault ID. The history file in which the fault entry was created is automatically selected as the current history file. A MATCH FAULT\_ID *fault\_id* command is issued so that only the newly created fault entry is displayed. This command is issued because the newly created fault entry might not be at the top of the chronologically ordered Fault Entry List display, and thus difficult to locate. To again see all fault entries in the history file, enter the MATCH ALL command (normally mapped to PF12).

The performance of subsequent reanalysis of the fault is improved by creating a fault history entry for a CICS system abend, and writing a minidump.

# Chapter 7. Formatting a CICS auxiliary trace data set

## Selecting a CICS auxiliary trace data set

To select a CICS auxiliary trace data set, first select the Format CICS Auxiliary Trace Data Set option from the Fault Entry List display File menu (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This opens the Format CICS Auxiliary Trace Data Set display as shown in Figure 116.

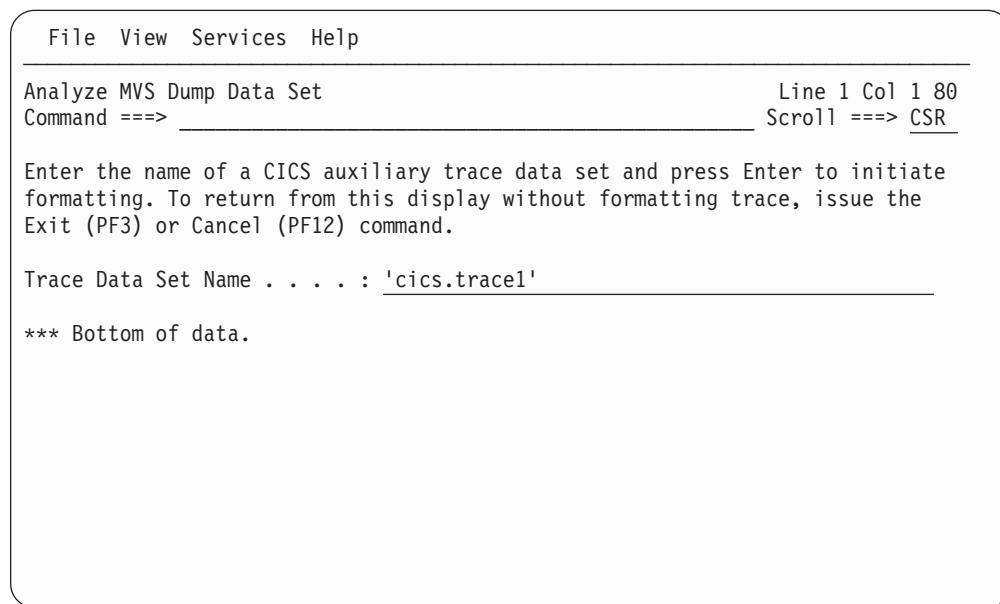


Figure 116. Sample Format CICS Auxiliary Trace Data Set display

On this display, type the name of the data set containing the CICS auxiliary trace to be formatted. The data set name-specification follows the ISPF data set name rules, that is, a data set name that is not enclosed in single quotes is prefixed by the current TSO profile prefix. The data set specified is checked for existence before being accepted. In this example, data set CICS.TRACE1 is being formatted.

The last data set name that is specified is stored in the ISPF profile for your application ID and is used for initialization of the display.

When the specified trace data set name has been validated, the Fault Analyzer CICS auxiliary trace formatting commences as indicated by the “Processing auxiliary trace data set. Please wait...” message being displayed (Figure 117 on page 188).

## Specifying CICS Trace Selection Parameters

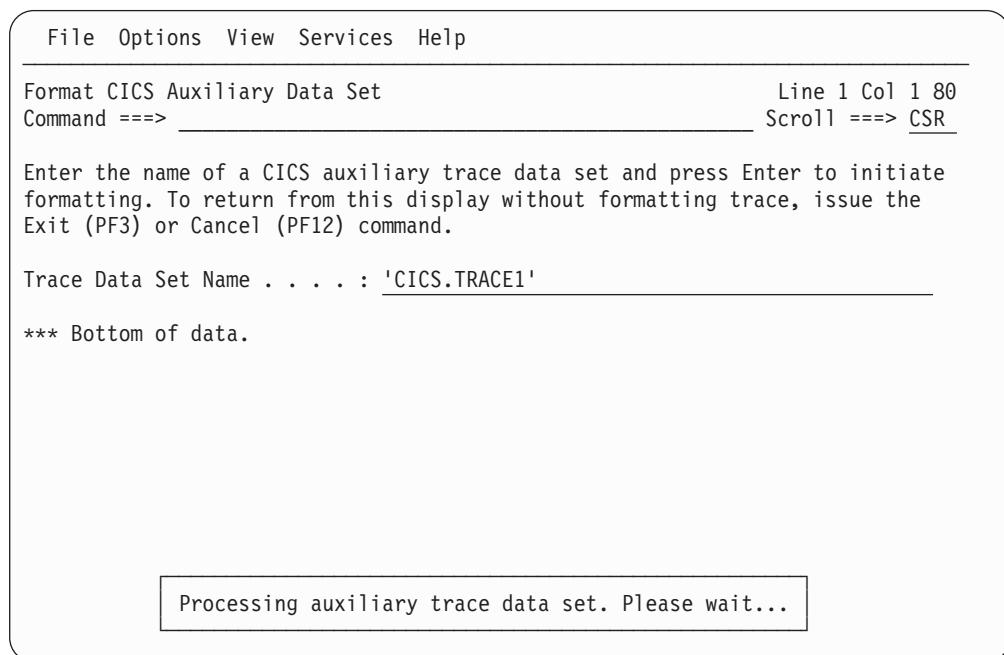


Figure 117. Recognizing that trace processing has commenced

## Specifying CICS Trace Selection Parameters

When the initial trace processing has ended, then the CICS Trace Selection Parameters display, of which an example is shown in “CICS Trace Formatting” on page 123, allows you to specify which internal trace entries are to be displayed and what level of formatting is required.

---

## Chapter 8. Performing Java analysis

A feature unique to the interactive component of Fault Analyzer is the ability to present information that is related to Java. The Java execution might be under WebSphere, CICS or Unix System Services on MVS. Typically, the environment is Java calling legacy programs. This chapter tells you how to:

- Set options for Java analysis
- Select a Java dump data set, or an existing Java fault entry, for analysis
- Display the resulting Java information in the interactive report

The steps outlined in the following assume that you have already started the interactive component of Fault Analyzer from an ISPF session.

**Note:** Java analysis is only compatible with the JVMs shown in Table 1 on page 10.

---

### Setting options for Java analysis

The general interactive reanalysis options are also used for Java analysis (see “Interactive reanalysis options” on page 103).

An optional IDIJVM DD statement added to the IDIS subsystem start-up JCL allows for a default JVM to be used for performing Java analysis (for details, see “IDIS subsystem requirements for Java”).

---

### Selecting a Java dump data set

Specification of a Java dump data set to be analyzed is done by first selecting the Analyze MVS Dump Data Set option from the Fault Entry List display File menu (for information about selecting this option in general, refer to “Action-bar pull-down menus” on page 61). This option brings up the Analyze MVS Dump Data Set display on which the MVS dump data set name can be entered. An example of this display is shown in “Selecting a CICS dump data set” on page 177.

If Java activity is detected in the dump being analyzed, then a history file fault entry is required to be created early while asynchronous DTFJ processing is performed. For this reason, the Create Java Fault Entry display is presented, as the example shown in Figure 118 on page 190.

## Selecting a Java dump data set

```
Create Java Fault Entry
Command ===> _____ Line 1 Col 1 76
                                                               Scroll ===> CSR

To create a fault entry for this Java dump, specify a history file data
set name and press Enter, or press PF3/PF12 to cancel the fault entry
creation and continue the analysis with incomplete Java information.

History file DSN. . . . : SWILKEN.HIST
*** Bottom of data.
```

Figure 118. Sample Create Java Fault Entry display

By specifying a valid history file to which the user has at least UPDATE access, and pressing the Enter key, a fault entry is created and the analysis continues with DTFJ processing performed asynchronously. However, Java information is not complete until the DTFJ processing has ended, thus requiring a subsequent reanalysis of the fault entry to be performed.

**Note:** Fault Analyzer Java DTFJ processing requires that the MVS post-dump exit IDIXTSEL is installed (for details, see “Installing the MVS post-dump exit IDIXTSEL” on page 313) and the IDIS subsystem is started (for details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239).

When exiting the interactive reanalysis report following the initial analysis, the specified history file is displayed and a MATCH automatically performed to show only the fault entry just created for this dump.

If PF3 or PF12 is pressed instead of the Enter key, then the early fault entry creation is skipped, but the analysis still continues. In this case, the dump analysis is performed as if no Java activity was detected, and the user is prompted to create a fault entry when exiting the analysis instead.

## Java fault entry reanalysis

Only fault entries that were created from analysis of Java dumps, or from real-time analysis of Java abends, include Java information in the reanalysis report.

**Note:** Information about real-time invocation of Fault Analyzer from Java applications is provided in “Invoking Fault Analyzer from Java catch block” on page 25 and “Invoking Fault Analyzer from Java dump events” on page 26.

Use the "I" line command against a history file fault entry to perform interactive reanalysis.

If the asynchronous DTFJ processing for the fault entry has not yet completed, then the Confirm Java Fault Entry Reanalysis display is presented, as the example shown in Figure 119.

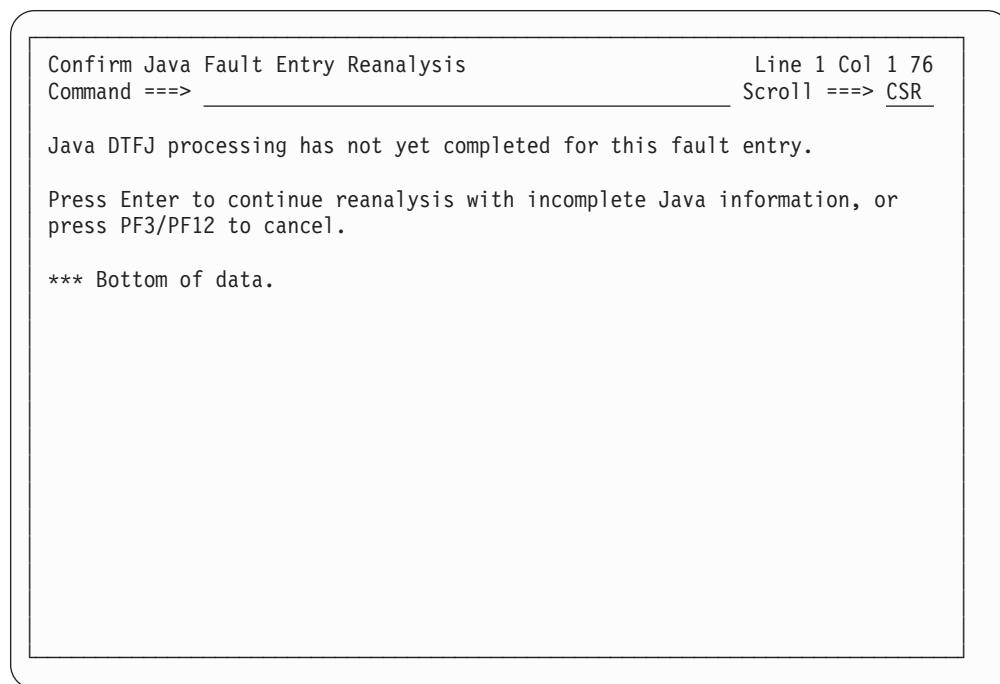


Figure 119. Sample Confirm Java Fault Entry Reanalysis display

By pressing the Enter key, reanalysis continues, but the Java information is incomplete.

If instead PF3 or PF12 is pressed, then the reanalysis is cancelled. Subsequent reanalysis at a later point in time might find that the asynchronous Java DTFJ processing has completed, and therefore continue the analysis without displaying this prompt.

If the asynchronous DTFJ processing for the fault entry failed to complete within a reasonable time, then message IDI0177E is issued instead of the Confirm Java Fault Entry Reanalysis display.

**Note:** If DTFJ processing never completes, then the reason might be one of the following:

- The MVS post-dump exit IDIXTSEL has not been installed (for details, see “Installing the MVS post-dump exit IDIXTSEL” on page 313).
- The IDIS subsystem has not been started (for details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239).
- The IDI.SIDIAUT2 data set has not been added to LINKLIST or provided via STEPLIB in the IDIS subsystem JCL (for details, see “IDIS subsystem requirements for Java” on page 244).

## Displaying the Java information in the interactive report

When the Java analysis has completed, the normal Interactive Reanalysis Report display is presented, as shown in Figure 120 on page 192.

## Displaying the Java information in the interactive report

```
File View Services Help
Interactive Reanalysis Report Line 1 Col 1 80
Command ===> Scroll ===> CSR
JOBNAME: BPXBATC3 SYSTEM ABEND: 0CB FAE3 2010/03/16 19:25:50
User Title: Java

Fault Summary:
Module /u/rturner/kenichiExample/j2c2cob/libMyDllLib.so, program MYCOB1,
offset X'3CE': Abend S0CB (Decimal-Divide Exception).

Select one of the following options to access further fault information:
1. Synopsis
2. Event Summary
3. Java Information
4. Storage Areas
5. Messages
6. Language Environment Heap Analysis
7. Abend Job Information
8. Fault Analyzer Options

{Fault Analyzer maximum storage allocated: 9.12 megabytes.}

*** Bottom of data.
```

Figure 120. Sample Java Interactive Reanalysis Report display

When selecting the Event Summary option from the Interactive Reanalysis Report display, both Java and non-Java events are included. An example is shown in Figure 121 on page 193.

## Displaying the Java information in the interactive report

File View Services Help							
Event Summary						Top of data Scroll ==> CSR	
Command ==>						FAE3 2010/03/16 19:25:50	
JOBNAME: BPXBATC3 SYSTEM ABEND: OCB							
{The following events are presented in chronological order.}							
Event #	Type	Fail Point	Module Name	Program Name	EP	Event Location (*)	Loaded
1	Call	BPXINLPA	n/a	n/a	M+49626		LPA
2	Call	n/a	n/a	n/a	n/a		Not de
3	Call	CEEINIT	n/a	CEEOPCM	E+8F8		LPA
4	>>> XPLink	CEEPLPKA	n/a	n/a	M+1BBB62		LPA
5	Call	java	JavaMain	JavaMain	P+34CC E+19EC		/apc/j
6	Java	n/a	n/a	Java2C2CobolExample.main	L#10		Not de
7	Call	libj9prt24.so	j9sig_protect	j9sig_protect	P+12DA E+5B2		/apc/j
8	Call	libj9vm24.so	gpCheckCallin	signalProtectAndRunGlue	P+29FC E+14		/apc/j
9	Call	libj9vm24.so	gpCheckCallin	gpProtectedRunCallInMethod	P+2C1A E+2A		/apc/j
10	Call	n/a	n/a	RUNCALLINMETHOD	n/a		Not de
11	<<< XPLink	CEEPLPKA	n/a	CEEVRONU	E+1026		LPA
12	Call	libMyDllLib.so	*Java_Ja	Java2C2CobolExample_callCobol	P+3F8 E+90		/u/rtu
13	Call	libMyDllLib.so	*Java_Ja	doSomething1	P+340 E+90		/u/rtu
14	Call	libMyDllLib.so	*Java_Ja	doSomething2			
15	Call	libMyDllLib.so	*Java_Ja	doSomething3	P+288 E+90 P+142 E+92		/u/rtu
16	Abend S0CB	*****	libMyDllLib.so	MYCOB1	MYCOB1 P+3CE E+3CE		/u/rtu
NOTE: Program names prefixed '*' are pseudo CSECT names created to help determine in which compile unit an entry point belongs.							
(*) One or more of the following abbreviations might appear in the "Event Location" column:							
F#n Source file number (refer to detailed event information for file identification)							
L#n Source file line number							
S#n Listing file statement number (refer to detailed event information for file identification)							
M+x Offset from start of load module							
P+x Offset from start of program							
E+x Offset from start of entry point							

Figure 121. Sample Java Event Summary display

## Displaying the Java information in the interactive report

Selecting the Java event # 6 from the above example, results in the Java event details display, as the example shown in Figure 122.

The screenshot shows a window titled "Java Event Details". At the top, there is a menu bar with "File", "View", "Services", and "Help". Below the menu is a toolbar with "Event Summary", "Command ==> \_\_\_\_\_", "Top of data", "Scroll ==> CSR", and "Help". The main content area is divided into sections:

- Previous Event Details**:  
This event occurred in Class Java2C2CobolExample Method main.  
Java source from /u/rturner/kenichiExample/j2c2cob/Java2C2CobolExample.java:  
Source  
Line #  
-5 }  
-4  
-3 public static void main(String args[]) {  
-2 System.out.println("Hello World from Java program!");  
-1 Java2C2CobolExample e = new Java2C2CobolExample();  
**000010** e.callCobol();  
+1 }  
+2 }  
+3  
+4
- Next Event Details**:  
The class static variable information is not available.  
The object instance variable information is not available.
- \*\*\* Bottom of data.

Figure 122. Sample Java Event Details display

Information specific to Java can be found by selecting the "Java Information" option from the Interactive Reanalysis Report display.

Selecting the Java Information point-and-shoot field from the above results in the presentation of the Java Information display, of which an example is shown in Figure 123 on page 195.

## Displaying the Java information in the interactive report

```
File View Services Help
Java Information Line 1 Col 1 80
Command ===> Scroll ===> CSR
JOBNAME: BPXBATC2 SYSTEM ABEND: OCB FAE2 2009/10/02 06:53:54

Java Version. . . . . : Java(TM) SE Runtime Environment(build 2.4).IBM
J9 VM(J2RE 1.6.0 IBM J9 2.4 z/OS s390-31
jvmmz3160-20080415_18762 (JIT enabled, AOT
enabled).J9VM - 20080415_018762_bHdSMr.JIT -
r9_20080415_1520.GC - 20080415_AA)

Java environment variables

- envvar

IBM_JAVA_COMMAND_LINE=java Java2C2CobolExample
LIBPATH=/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s390/j9vm:/apc/java631-UK
1/usr/lpp/java/J6.0/..../lib/s390:/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s
31-UK18621/usr/lpp/java/J5.0/bin:/apc/java531-UK18621/usr/lpp/java/J5.0/bin/
/j9vm:/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s390
_BPX_SHAREAS=NO
JAVA_HOME=/apc/java531-UK18621/usr/lpp/java/J5.0
HOME=/u/rturner
LOGNAME=RTURNER
SHELL=/bin/sh
_IDIZZDBG_ZFS=1
_CEE_RUNOPTS=POS(ON),TERMTHDACT(UAIMM),TRAP(ON,NOSPIE)
_CREATE_LAYOUT=Y
JAVA_DUMP_OPTS=ONANYSIGNAL(ALL)
TZ=UTC0
CLASSPATH=.:./u/rturner/kenichiExample/j2c2cob
_BPXK_MDUMP=./RTURNER.JAVA.MDUMP01
_EDC_PTHREAD_YIELD=-2
STEPLIB=RTURNER.A0.LOAD
PATH=/apc/java531-UK18621/usr/lpp/java/J5.0/bin/../../apc/java631-UK35911/usr/
=/apc/java631-UK35911/usr/lpp/java/J6.0/bin/java
```

Figure 123. Sample Java Information display (Part 1 of 2)

```

Java VM init args
- args
arg=-Xjcl:jclscar_24
arg=-Dcom.ibm.oti.vm.bootstrap.library.path=/apc/java631-UK35911/usr/lpp/jav
arg=-Dsun.boot.library.path=/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s390
arg=-Djava.library.path=/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s390:/apc
c/java631-UK35911/usr/lpp/java/J6.0/lib/s390:/apc/java631-UK35911/usr/lpp/ja
a/J6.0/lib/s390:::/u/rturner/kenichiExample/j2c2cob:/apc/java531-UK18621/u
va/J5.0/bin/j9vm:/apc/java631-UK35911/usr/lpp/java/J6.0/lib/s390/j9vm:/apc/j
arg=-Djava.home=/apc/java631-UK35911/usr/lpp/java/J6.0
arg=-Djava.ext.dirs=/apc/java631-UK35911/usr/lpp/java/J6.0/lib/ext
arg=-Duser.dir=/u/rturner
arg=_j2se_j9=71168
arg=-Xdump
arg=-Djava.class.path=.:./u/rturner/kenichiExample/j2c2cob
arg=-Dsun.java.command=Java2C2CobolExample
arg=-Dsun.java.launcher=SUN_STANDARD
arg=_port_library

Java threads with traceback information

Call trace for thread: main

Method Location
Java2C2CobolExample.callCobol Native Method
Java2C2CobolExample.main Java2C2CobolExample.java:10

Call trace for thread: Signal Dispatcher

Method Location
com.ibm.misc.SignalDispatcher.waitForSignal Native Method
com.ibm.misc.SignalDispatcher.run SignalDispatcher.java:66

```

*Figure 124. Sample Java Information display (Part 2 of 2)*

---

# Chapter 9. The Fault Analyzer report

This chapter describes the contents of the report that is written by Fault Analyzer.

---

## General report information

The analysis report produced by batch reanalysis has the same structure as the real-time report. The interactive reanalysis report has a similar structure, however, you are able to select a particular section for viewing, rather than having to view the entire report in sequence. In addition, the interactive report provides specific information that is related to CICS system abends or Java analysis, that is not included in the real-time or batch analysis reports. For further details, see Chapter 6, "Performing CICS system abend dump analysis," on page 177.

The Fault Analyzer report is written to DDname IDIREPRT for real-time analysis, or to SYSPRINT for batch reanalysis.

### Most significant abend code

If the analysis of a fault results in more than one abend code being presented, then Fault Analyzer always considers the first abend code (chronologically) the most significant. However, if the Language Environment CEEWUCHA user condition handler is used (registered via the LE runtime option USRHDLR(CEEWUCHA)), then the final or last abend code is usually the one that provides the most detailed information about the error.

### Open file record information

The records for open sequential files are shown in chronological order.

For input files, a 'previous—current—next' sequence is used with the current record being the record that was provided for application use prior to the fault.

For output files, "last record written" and "current record build area" descriptions are used.

### Spanned record interpretation

When viewing the open file record information, where variable spanned records might be present, the user is required to reassemble the logical view of the record if a spanned record is involved. This requirement is because the buffers on which the record information is based might not still contain all of the segments for a particular spanned record. When spanned records are involved, the record information heading for each record contains "first segment", "intermediate segment", and "last segment" according to the data available and the structure of the particular spanned record (there might be zero to many intermediate segments).

Following are two real-time report extracts of open file information.

Figure 125 on page 198 shows file information and buffers for an input file:

## General report information

---

### Open Files

```
File Name . . . . . : INDD
  Data Set Name . . . . . : JERRYBL.OUT80S
  File Attributes . . . . . : ORGANIZATION=SEQUENTIAL, ACCESS MODE=SEQUENTIAL,
                             RECFM=VARIABLE BLOCKED SPANNED
  Last I/O Function . . . . : READ
  Open Status . . . . . : INPUT
  File Status Code. . . . . : 0

  Previous Record -2. . . . : Segment data length 6, variable record first segment
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F32           C6C6C6F6 F6F6          *FFF666      *

  Previous Record -1. . . . : Segment data length 32, variable record intermediate segment
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F40           40404040 40404040 40404040 40404040 *          *
  Line 08053F50 same as above

  Previous Record . . . . . : Segment data length 2, variable record last segment
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F68           4040          *          *

  Current Record. . . . . : Record data length 20, variable record
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F6E           C7C7C7F7 F7F74040 40404040 40404040 *GGG777      *
  08053F7E           +10 40404040          *          *

  Next Record . . . . . : Segment data length 2, variable record first segment
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F86           C8C8          *HH         *

  Next Record +1. . . . . : Segment data length 8, variable record last segment
  Address Offset      Hex          EBCDIC
  ----- -----
  08053F90           C8F8F8F8 40404040          *H888      *

NOTE: Some segments not available due to buffer wrap-around.
```

---

Figure 125. Sample real-time report extract of open input file information

Figure 126 on page 199 shows file information and buffers for an output file:

## Open Files

```

File Name . . . . . : OUTDD
Data Set Name . . . . . : JERRYBL.OUT80S
File Attributes . . . . . : ORGANIZATION=SEQUENTIAL, ACCESS MODE=SEQUENTIAL,
                           RECFM=VARIABLE BLOCKED SPANNED
Last I/O Function . . . . : WRITE
Open Status . . . . . : OUTPUT
File Status Code. . . . . : 0

Last Record Written -2. . : Segment data length 20, variable record first segment
Address Offset   Hex          EBCDIC
-----
08053F74        C9C9C9F9 F9F94040 40404040 40404040 *III999      *
08053F84        +10 40404040          *          *          *

Last Record Written -1. . : Segment data length 32, variable record intermediate segment
Address Offset   Hex          EBCDIC
-----
08053F90        40404040 40404040 40404040 40404040 *          *
Line 08053FA0 same as above

Last Record Written . . . : Segment data length 28, variable record last segment
Address Offset   Hex          EBCDIC
-----
08053FB8        40404040 40404040 40404040 40404040 *          *
08053FC8        +10 40404040 40404040 40404040          *          *

Current Record Build Area : RDW is zero, no length assigned yet
Address Offset   Hex          EBCDIC
-----
08053E90        D1D1D1C1 C1C14040 40404040 40404040 *JJJAAA      *
08053EA0        +10 40404040 40404040 40404040 40404040 *          *
Lines 08053EB0-08053EC0 same as above
08053ED0        +40 40404040 40404040 40404040 40406E6E *          >>*
08053EE0        +50 00000000          *....          *          *

```

Figure 126. Sample real-time report extract of open output file information

## COBOL suppressed copybooks

When large copybooks are included in COBOL programs, the SUPPRESS option is often used to stop the expansion of the copybook in the compile listing. For example:

```
COPY copy-book-name SUPPRESS
```

To include the suppressed copybooks in the Fault Analyzer batch report working storage map (whether real-time analysis or reanalysis), it is necessary to specify the Detail(Long) option.

Suppressed copybook information is always available for selection in the interactive reanalysis report.

**Note:** Suppressed copybook information might not always be complete, or totally accurate, because it is a 'best attempt' at rebuilding information that was suppressed from the compiler listing.

## Main report sections

The following describes each of the main report sections.

## Main report sections

### The prolog section

The prolog section consists of everything from the top of the report until the start of the synopsis section.

At the top of the prolog section is information about the version, release, and modification level of Fault Analyzer, as well as the latest ++APAR or PTF installed. Following this information is the Fault Analyzer copyright statement and a header for the report.

If performing batch reanalysis of a dump data set, or a fault entry created from interactive reanalysis of a dump data set, and the dump data set or fault entry contains CICS system or Java environment information, then a note is added here. This note tells you that the better way to analyze the current fault is by using the appropriate ISPF interface reanalysis method. This approach is better because Fault Analyzer for these types of environments provides specific support in the interactive reanalysis report which enables the user to see information that is not included in the batch report.

### The synopsis section

The synopsis section provides a brief description of the fault and its analysis. It is preceded by the heading:

```
<H1> I B M   F A U L T   A N A L Y Z E R   S Y N O P S I S
```

### The summary section

The summary section contains a list of all events (such as abends and call entries) in chronological order. It is preceded by the heading:

```
<H1> I B M   F A U L T   A N A L Y Z E R   E V E N T   S U M M A R Y
```

With each event entry in the list is also provided key information, such as module name, program name, and failing line or statement number. This information is a summary of the information that is provided in the details section for the event.

The following is provided for each entry in the list:

#### Event #

This section is a unique sequence number in ascending chronological order assigned to the event. The event number is used as reference to the detailed information for the event in the Event Details section of the report.

#### Event Type

The event type as one of the following:

##### Abend *abend-code*

An abend event.

##### BALR

A BALR event.

##### Call

A CALL event.

##### Current PRB

A current PRB event.

##### Debug Tool

A Debug Tool for z/OS event.

##### EXEC *type*

An EXEC event.

<b>Execute</b>	An EXECUTE event.
<b>IDISNAP</b>	An IDISNAP event.
<b>Interrupt</b>	An interrupt event.
<b>Java</b>	A Java event.
<b>LE Condition</b>	An Language Environment condition event.
<b>Link</b>	A LINK event.
<b>ONCODE code</b>	A PL/I ONCODE event.
<b>Prog Int</b>	A program interruption event.
<b>SVC number</b>	An SVC event.
<b>Fail Point</b>	If the event has been identified as the point of failure, then this identification is indicated by ***** in this column.
<b>Module Name</b>	If available, the name of the load module that is associated with the event. Otherwise, n/a.
<b>Program Name</b>	If available, the name of the program or CSECT that is associated with the event. Otherwise, n/a.
<b>EP Name</b>	If available, the name of the entry point that is associated with the event. Otherwise, n/a.
<b>Event Location</b>	One or more of the following abbreviations might appear in this field:
<b>F#n</b>	Source file number
<b>L#n</b>	Source file line number
<b>S#n</b>	Listing file statement number
<b>M+x</b>	Offset from start of load module
<b>P+x</b>	Offset from start of program
<b>E+x</b>	Offset from start of entry point
<b>Description</b>	The following information is provided: <ul style="list-style-type: none"> <li>• If available, a brief description of the program or load module function.</li> <li>• The data set name from where the module was loaded, or LPA if the module was found in LPA, or CSA if the module was found in CSA.</li> </ul> If the data set name cannot be determined for a module not in LPA or CSA, then Not determined is shown.

## Main report sections

### Maximum call depth

The maximum number of events that can be included in the event summary is 200. If the fault contains more than 200 events, then only the first 100 and the last 100 events are shown, with an indication of the events which have been suppressed due to the maximum call depth having been exceeded.

### The event details section

The event details section provides detailed information about each event. It is preceded by the heading:

```
<H1> I B M   F A U L T   A N A L Y Z E R   E V E N T   D E T A I L S
```

The types of events included in the details section is subject to the Detail option in effect. Included in the detailed event section is also more information that is associated with the event, such as message descriptions (extracted by Fault Analyzer so that you do not need to look this up in a manual) and the contents of the program's working storage. When appropriate, you also find information such as abend code explanations and open file buffers here.

Source code information that is shown in the details section of the report includes up to 5 source lines or statements ahead of, and following, the source location for the event. If the event source location is within an expanded assembler macro, then the extra source statements are in addition to the statements caused by the macro expansion.

To change the default extra five lines or statements, use the Detail option (see “Detail” on page 465) or the Analysis Control user exit (see “Analysis Control user exit” on page 368).

If no matching compiler listing or side file was provided for the point-of-failure event, then the failing machine instruction is shown. In addition, up to 12 instructions ahead of, and 6 following, the failing instruction are provided. Here example listing. It is based on the IDIVPCOB COBOL IVP, but with no compiler listing or side file provided:

```
<H2> EVENT 1 OF 1: ABEND SOC7

*****
* P O I N T   O F   F A I L U R E *
*****

Abend Code. . . . . : SOC7
Program-Interruption Code . : 0007 (Data Exception)
                           A decimal digit or sign was invalid.

Most recently referenced data items:
  Data Item . . . . . : BLW=00000+018
    At Address. . . . . : 167900A0
    Length. . . . . . : X'4'
  Data Item Storage . . . : 0986888F  *.fh.*

  Data Item . . . . . : BLW=00000+020
    At Address. . . . . : 167900A8
    Length. . . . . . : X'4'
  Data Item Storage . . . : C1C2C3C4  *ABCD*
```

NOTE: Source code information could not be presented because the search for a compiler listing or side-file was unsuccessful for program IDISCBL1.

```
Load Module Name. . . . . : SYS05291.T124505.RA000.IDIVPCOB.GOSET.H02(IDISCBL1)
  At Address. . . . . : 16700D88
  Load Module Length. . . . : X'1278'
  Link-Edit Date and Time . : 2005/10/18 12:45:07

Program and Entry Point Name: IDISCBL1
  At Address. . . . . : 16700D88 (Module IDISCBL1 offset X'0')
```

```
Program Length. . . . . : X'638'
Program Language. . . . . : COBOL (Compiled using COBOL for OS/390 & VM V2 R2
                           M2 on 2005/10/18 at 12:45:06)
```

```
Machine Instruction . . . . . : FD73D0B8D0A8 DP 184(8,R13),168(4,R13)
At Address. . . . . . . . . . : 1670115C (Program IDISCB1 offset X'3D4')
AMODE . . . . . . . . . . . . : 31
Failing Operand . . . . . . . : Second operand
First Operand Address . . . . . : 0002A0D0 (171824 bytes of storage addressable)
First Operand Length. . . . . : 8
First Operand Storage . . . . . : 00000000 0986888C *....fh.**
Second Operand Address. . . . . : 0002A0C0 (171840 bytes of storage addressable)
Second Operand Length . . . . . : 4
Second Operand Storage. . . . . : C1C2C3CF *ABC.*
```

Instructions around point of failure:

Offset	Hex	Instruction	
-36	D203 D094 D098	MVC 148(4,R13),152(R13)	
-30	5820 905C	L R2,92(,R9)	
-2C	58F0 202C	L R15,44(,R2)	
-28	4110 A0E9	LA R1,233(,R10)	
-24	05EF	BALR R14,R15	
-22	58B0 C014	L R11,20(,R12)	
-1E	47F0 B1CC	BC 15,460(,R11)	
-1A	D203 D0A8 8020	MVC 168(4,R13),32(R8)	BLW=00000+020
-14	960F D0AB	OI 171(R13),15	
-10	D203 D0B0 8018	MVC 176(4,R13),24(R8)	BLW=00000+018
-A	960F D0B3	OI 179(R13),15	
-6	F873 D0B8 D0B0	ZAP 184(8,R13),176(4,R13)	
*****	FD73 D0B8 D0A8	DP 184(8,R13),168(4,R13)	
+6	D201 8028 D0BA	MVC 40(2,R8),186(R13)	BLW=00000+028
+C	940F 8028	NI 40(R8),15	BLW=00000+028
+10	960F 8029	OI 41(R8),15	BLW=00000+029
+14	5830 D094	L R3,148(,R13)	
+18	07F3	BCR 15,R3	
+1A	9120 9054	TM 84(R9),32	

Program Status Word (PSW) . . . : 078D2000 96701162

#### General Purpose Registers:

```
R0: 0002A0D8 (171816 bytes of storage addressable)
R1: 16700F91 (Module IDISCB1 program IDISCB1 + X'209')
R2: 0001B7FC (231428 bytes of storage addressable)
R3: 16701126 (Module IDISCB1 program IDISCB1 + X'39E')
R4: 16700DC0 (Module IDISCB1 program IDISCB1 + X'38')
R5: 00016AF0 (251152 bytes of storage addressable)
R6: 00000000 (2048 bytes of storage addressable)
R7: 00000000 (2048 bytes of storage addressable)
R8: 16790088 (Module IDISCB1 program IDISCB1 WORKING-STORAGE SECTION
               BLW=00000 + X'0')
R9: 1678C100 (253696 bytes of storage addressable)
R10: 16700E9C (Module IDISCB1 program IDISCB1 + X'114')
R11: 16700FBC (Module IDISCB1 program IDISCB1 + X'234')
R12: 16700E84 (Module IDISCB1 program IDISCB1 + X'FC')
R13: 0002A018 (172008 bytes of storage addressable)
R14: 967010D2 (Module IDISCB1 program IDISCB1 + X'34A')
R15: 96759E60 (Module IGZCPAC + X'3C348')
```

## The system-wide information section

This section contains, for example, console messages that are not identified as belonging to any specific event, or CICS system-related information, such as trace data and 3270 screen buffer contents. It is preceded by the heading:

<H1> S Y S T E M - W I D E I N F O R M A T I O N

Information about open files that could not be associated with any specific event might also be included here. If there is no information in this section, then it does not appear in the report.

## Main report sections

### The abend job information section

This section provides information about the abending job that is associated with the real-time invocation of Fault Analyzer or, in the case of reanalysis, the minidump or MVS dump analyzed. It is preceded by the heading:

```
<H1> I B M   F A U L T   A N A L Y Z E R   A B E N D   J O B   I N F O
```

### The options section

This section is a list of the Fault Analyzer options that were in effect at the time of the analysis. It is preceded by the heading:

```
<H1> I B M   F A U L T   A N A L Y Z E R   O P T I O N S
```

### The epilog section

The epilog section consists of everything from the bottom of the options section until the end of the report.

For a real-time analysis report, information is provided about the invocation exit used and the approximate amount of above-the-line storage that is allocated during the analysis, followed by the fault ID assigned. A batch reanalysis report instead provides information about the fault ID and history file that is used.

The last line of the report provides information about the time and date when the report was created.

---

## Sample reports

Sample Fault Analyzer reports are provided in the IDI.SIDIDOC1 data set as shown in the following.

*Table 5. Sample reports*

Description	IDI.SIDIDOC1 member name
COBOL IVP. This report was produced by running the Fault Analyzer supplied COBOL installation verification program (IVP), provided as member IDIVPCOB in data set IDI.SIDISAM1.	IDISRP01
PL/I IVP. This report was produced by running the Fault Analyzer supplied PL/I installation verification program (IVP), provided as member IDIVPPLI in data set IDI.SIDISAM1.	IDISRP02
Assembler IVP. This report was produced by running the Fault Analyzer supplied assembler installation verification program (IVP), provided as member IDIVPASM in data set IDI.SIDISAM1.	IDISRP03
C DB2 IVP. This report was produced by running the Fault Analyzer supplied C DB2 installation verification program (IVP), provided as member IDIVPDB2 in data set IDI.SIDISAM1.	IDISRP04
MQSeries. This report was produced by performing reanalysis of a fault entry created for an MQSeries abend.	IDISRP05
C IVP. This report was produced by running the Fault Analyzer supplied C installation verification program (IVP), provided as member IDIVPC in data set IDI.SIDISAM1.	IDISRP06

---

## **Chapter 10. Using non-ISPF interfaces to access Fault Analyzer history files**

The following describes different ways to access Fault Analyzer history files as alternatives to using TSO/ISPF.

---

### **Using the Fault Analyzer plug-in for Eclipse**

A Fault Analyzer Eclipse plug-in is available for use with CICS Explorer. This plug-in communicates with a Fault Analyzer z/OS server job using TCP/IP to allow the viewing of history file fault entries.

This plug-in can be used with CICS Explorer as an alternative to the Fault Analyzer RDz client described in "Using the Fault Analyzer client for IBM Rational Developer for System z."

Details on using the Fault Analyzer plug-in can be found in the help of CICS Explorer. In the action bar click on "Help", and scroll down and select "Help Contents". In the subsequent help window there is a section called "Fault Analyzer for z/OS client for Eclipse".

Refer to "Installing the Fault Analyzer plug-in for Eclipse" on page 443 for information about installing this interface.

---

### **Using the Fault Analyzer client for IBM Rational Developer for System z**

If you already have IBM Rational Developer for System z (RDz) installed, then a Fault Analyzer client is available from RDz for this environment, which provides functionality equivalent to that of the Eclipse plug-in described in "Using the Fault Analyzer plug-in for Eclipse."

The Fault Analyzer client for IBM Rational Developer for System z provides the following features:

- An interface to fault history files and views from an IBM Rational Developer for System z environment.
- The ability to work with multiple fault history files and views from multiple systems.
- The ability to browse fault entries that were created during real-time analysis of abending programs.
- A browser for browsing the dump storage that is associated with a fault entry.
- A source listing of abending programs on demand using side files.

Refer to [http://www.ibm.com/developerworks/rational/library/07/1218\\_yoshimura-faydi/](http://www.ibm.com/developerworks/rational/library/07/1218_yoshimura-faydi/) for more information about using this interface.

### Performing interactive reanalysis under CICS

Fault Analyzer uses a special component to display ISPF panels that can allow it to operate as a CICS transaction to view history files and perform interactive reanalysis. This capability under CICS does not use TSO—it is intended for users who might not have TSO logon capability on an MVS image, but have a need to review and analyze history file information on that MVS image.

The capabilities of Fault Analyzer running as a CICS transaction are almost identical to Fault Analyzer under TSO/ISPF (as described in Chapter 3, “The Fault Analyzer ISPF interface,” on page 33), with the following restrictions and variations:

1. Batch reanalysis of the fault entry is not supported.
2. Functions which invoke ISPF EDIT are not supported:
  - Editing the options data set prior to interactive reanalysis.
  - Altering allocated data sets prior to interactive reanalysis.
  - EDIT a User Formatting EXEC from list of available EXECs.
  - EDIT of a DSECT from the DSECT list display.
3. Since this component is not running under TSO, no prefixing of data set names is performed. That is, where a data set name can be entered, for example a Fault History File or View, the data set name needs to be fully qualified with or without quotes.
4. ISPF profile changes made while in the Interactive Reanalysis Report, for example changing the location of the command line, might not be immediately reflected upon return to the Fault Entry List display. However, the profile changes are detected on the next invocation of the main CICS transaction.

Refer to “Enabling interactive reanalysis under CICS” on page 445 for information about installing this interface.

---

### Using the Fault Analyzer web interface

A web interface for Fault Analyzer is provided as an alternative to the Eclipse plug-in, without the need for installing any software on a user's computer. Refer to “Installing the Fault Analyzer web interface” on page 446 for information about installation of this interface. Once installed, the Fault Analyzer web interface can be viewed by opening a web browser and navigating to the URL:

`http://<HostName>:9088/IDIGWEB/`

**Note:** The default port is set to 9088 but the value can be modified as described in “Customizing the server configuration” on page 447.

A sample Fault Analyzer web interface display is provided in Figure 127 on page 207.

## using the Fault Analyzer web browser interface

The screenshot shows the Fault Analyzer web browser interface. At the top, there's a navigation bar with tabs for 'Fault Analyzer' and 'Fault Options (F00024)'. On the right of the bar are buttons for 'ALPOGA', a question mark icon, and 'IBM.'. Below the bar, on the left, is a sidebar with 'History Files' and 'View Files' buttons, and a dropdown menu for selecting a history file. The main content area displays a fault summary for 'F00024'. It includes sections for 'Summary Information' (showing a module error), 'Synopsis' (describing a system abend), and source code (a C division operation). At the bottom of the main content area is a navigation bar with tabs for 'Summary', 'Events', 'Job Info', 'System Info', 'Report Info', 'User notes', and 'User bookmarks'. Below this is a table titled 'No filter applied' showing a list of faults. The table has columns for 'FAULT\_ID', 'JOB/TRAN', 'USER\_ID', and 'SYS/JOB'. The data in the table is as follows:

FAULT_ID	JOB/TRAN	USER_ID	SYS/JOB
F75775	AS670F1/CECI	SIMCOCK	FAE1/AS670F1
F75774	AS670F1/CECI	SIMCOCK	FAE1/AS670F1
F00018	MYFIRST	ALPOGA	FAE1/MYFIRST
F00020	MYFIRST	ALPOGA	FAE1/MYFIRST

At the bottom of the page, there's a footer bar with buttons for navigating between pages.

*Figure 127. Sample Fault Analyzer web interface display*

The following web browsers are supported:

- Firefox 10 ESR+
- Safari 6.0+ - Mac
- Safari 5.1+ - Win
- Google Chrome
- Internet Explorer 8+

For an overview of how to use the interface, click the Help information link in the web interface help menu (the question mark button), found in the top-right hand corner of the display.



---

## **Part 2. Fault Analyzer installation and administration**



---

## Chapter 11. Migrating from an earlier version of Fault Analyzer

Information about migrating from an earlier version of Fault Analyzer is provided in the following.

**Note:** The information in “LPA module compatibility” on page 216 and “Sharing of history files across a sysplex with mixed levels of Fault Analyzer” on page 216 is applicable to all versions of Fault Analyzer.

See Chapter 29, “Maintaining Fault Analyzer,” on page 353 for information about the SMP/E installation and subsequent activation of the new version of Fault Analyzer.

---

### Migrating from V12.1 to V13.1

This section provides information about changes to Fault Analyzer version 13.1 that you should be aware of if migrating from version 12.1.

- The following Fault Analyzer SMP/E ++USERMODs are no longer available:  
IDILEDS  
IDISCNF  
IDISRFR  
IDISXCUM

These have been replaced by equivalent settings in the IDIOPTLM configuration-options module. For details, see Chapter 17, “Customize Fault Analyzer by using an IDIOPTLM configuration-options module,” on page 257.

- If the use of IDIOPTLM is required to specify an alternative data set for IDICNFxx, then the use of the IBM Problem Determination Tools for z/OS Common Component IPVOPTLM configuration-options module is similarly required for access to IPVCNF00. Otherwise, abend S913 and message ICH408I (or similar, depending on the security server used) might occur due to insufficient READ access to either parmlib member. For more information, see “Specifying an alternative parmlib data set for IDICNFxx” on page 258.
- The Fault Analyzer Locale option *locale-name* can now alternatively be specified in the IBM Problem Determination Tools for z/OS Common Component IPVCNF00 parmlib member. For more information, see “Locale” on page 480.
- The HistCols and InteractiveExitPromptSeconds options have been deprecated and replaced by the FAISPFOpts option. For details, see “FAISPFOpts” on page 475.
- The SVC dump registration exit IDIXTSEL is no longer an optional installation item, but is required. For details, see “Checklist for installing and customizing Fault Analyzer” on page 217.
- Java dump capture requires access to the IDI\_SDUMP\_ACCESS XFACILIT profile. For details, see “Invoking Fault Analyzer from Java catch block” on page 25.
- It is a requirement for CICS open TCB users to include the Fault Analyzer IDIPLT program in the CICS startup PLT. For details, see “Adding the required programs to the startup PLT” on page 303.
- The UseIDISTime option has been deprecated and is now always in effect. For details, see “UseIDISTime” on page 503.

## Migrating from V12.1 to V13.1

- The Fault Analyzer plug-in for Eclipse is no longer provided as member IDIGUIP in data set IDLSIDIDOC2. However, the plug-in is still available via the web, as explained in “Download Fault Analyzer plug-in” on page 444.
- Changes have been made to the IDIXFXIT user exit parameter list:
  - The sixth parameter was previously a pointer to the fault entry ID for which the exit was invoked. Now, the sixth parameter is the address of an HD segment data area, which includes not only the fault entry ID, but also various other information that is related to the fault entry.
  - The security server user and default group ID, provided as the second and third parameters, are now always those of the exit caller, whereas previously they were at times obtained from the fault entry itself. Note that the HD segment data area now pointed to by the sixth parameter includes the fault entry creator security server user and default group ID information, when available.

For details, see “Using the IDIXFXIT user exit” on page 276.

- The current value in ENV.VERSION has changed to 0005. This is due to ENV.FORMATMING\_EXIT having changed from always specifying 'C' for a CICS transaction fault, to now specifying 'C' if Fault Analyzer was invoked via the XPCABND exit or 'D' if invoked via the XDUREQ exit.

---

## Migrating from V11.1 to V12.1

This section provides information about changes to Fault Analyzer version 12.1 that you should be aware of if migrating from version 11.1.

- The IBM Problem Determination Tools for z/OS Common Component must be installed in order for Fault Analyzer to provide source level support when using compiler listings or SYSADATA files.
- BookManager softcopy books are no longer shipped with Fault Analyzer. As a result, the Fault Analyzer ISPF interface "Help->Fault Analyzer User's Guide and Reference..." action-bar pull-down option has been removed.
- The following user exit data area fields have been removed:
  - EPC.MINIDUMP\_PAGES (replaced by ENV.MINIDUMP\_PAGES)
  - UFM.NUM\_FPREGS (use UFM.FPREG0 through UFM.FPREG15 instead)
  - UFM.FPREG\_DATA\_ADDRESS (use UFM.FPREG0 through UFM.FPREG15 instead)

---

## Migrating from V10.1 to V11.1

This section provides information about changes to Fault Analyzer version 11.1 that you should be aware of if migrating from version 10.1.

- To enable Java analysis, data set IDISIDIAUT2 must be added as STEPLIB in the IDIS subsystem start-up JCL. For details, see “Starting the IDIS subsystem” on page 241.
- Users of Message and Abend Code Explanation load module user exits should note that the offset to the XPL.ABEND\_CODE field has changed.
- In the past, prompting for missing compiler listings or side files during interactive reanalysis depended on whether any compiler listing or side file data sets had been provided to Fault Analyzer, implicitly or explicitly, by the time when these were required during the reanalysis of a fault entry. As a result, prompting could occur for some fault entries, but not for others.

With the user-specified option now available to control prompting (see “Interactive reanalysis options” on page 103), it is easier to ensure a consistent

behavior. If users have not yet set this option explicitly, then the default setting can be controlled by one of these actions:

- Specify one or more compiler listing or side file data sets in the IDICNFxx parmlib member (in which case the default is to prompt).
- Ensure that no such data sets are specified in the IDICNFxx parmlib member (in which case the default is to not prompt).

---

## Migrating from V9.1 to V10.1

This section provides information about changes to Fault Analyzer version 10.1 that you should be aware of if migrating from version 9.1.

- The IDIJ subsystem is no longer used.

---

## Migrating from V8.1 to V9.1

This section provides information about changes to Fault Analyzer version 9.1 that you should be aware of if migrating from version 8.1.

- The Analysis Control user exit is now being invoked in reanalysis mode, as well as in real time.

Existing Analysis Control user exits should be reviewed to ensure that they are appropriate for use in reanalysis mode also. Assignment of history file is ignored if not real time.

- The following fields have been removed from the EPC user exit data area:
  - EPC.DUPLICATE\_COUNT (replaced by ENV.DUPLICATE\_COUNT)
  - EPC.POF\_CSECT\_NAME (replaced by ENV.POF\_CSECT\_NAME)
  - EPC.POF\_CSECT\_OFFSET (replaced by ENV.POF\_CSECT\_OFFSET)
  - EPC.POF\_MODULE\_LKED\_DATE (replaced by ENV.POF\_MODULE\_LKED\_DATE)
  - EPC.POF\_MODULE\_LKED\_TIME (replaced by ENV.POF\_MODULE\_LKED\_TIME)
  - EPC.POF\_MODULE\_NAME (replaced by ENV.POF\_MODULE\_NAME)
- The ENV.LOCK\_FLAG field size is now two characters instead of one, with support for fault entry expiration control—for details, see “Fault entry expiration control” on page 61. Users of load module user exits should note that the field offset has changed,
- The current value in ENV.VERSION has changed to 0004.
- Users of load module user exits should note that the total size of the ENV data area has increased.
- The WDZClient option has been renamed to RDZClient. However, the WDZClient option is still supported for compatibility.

---

## Migrating from V7.1 to V8.1

This section provides information about changes to Fault Analyzer version 8.1 that you should be aware of if migrating from version 7.1.

- BookManager softcopy books are no longer shipped with Fault Analyzer. (They provided message and abend code explanations.) Instead, a VSAM file is populated with this information.

Related changes include:

- Allocating and populating the new VSAM cluster. For details, see “Setting up the message and abend code explanation repository” on page 233.
- Deletion of the old cache data set to reclaim DASD space.

## Migrating from V7.1 to V8.1

- Removal of any IDICACHE suboption specifications of the DataSets option. These are likely to be found in the IDICNFxx parmlib member.

**Note:** While the specification of this suboption is currently ignored, it should be removed to prevent possible error messages from being issued in the future.

Once the IDICACHE data set is no longer required for use with Fault Analyzer V7.1, it can be deleted.

- The Batch Report Tailoring user exit support has been removed.

Related changes include:

- Removal of any REPORT suboption specifications of the Exits option. These are likely to be found in the IDICNFxx parmlib member.

**Note:** While the specification of this suboption is currently ignored, it should be removed to prevent possible error messages from being issued in the future.

- If previously, extra source lines had been requested for real-time or batch reanalysis reports by setting the REP.EXTRA\_SOURCE\_LINES field to a positive value using a Batch Report Tailoring user exit, then you now instead need to use the Detail option (see “Detail” on page 465) or the Analysis Control user exit (see “Analysis Control user exit” on page 368).

- The high-level qualifier of the default recovery fault recording data set name has changed. Previously, this was IDIDUMP, now it is IDIRFRHQ. If your installation relies on the default name, then it might be necessary to change security server profiles to ensure that users are still able to allocate data sets with the new name (see “Managing recovery fault recording data set access” on page 234).

- The NoDup(ImageFast(0)) default option has changed to NoDup(ImageFast(5)). If IMS fast duplicate detection is not desired, then it is necessary to specify the NoDup(ImageFast(0)) option. For details, see “NoDup” on page 482.

- The IDIS subsystem PARM field options UPDINDEX and IMAGEFAST have become defaults.

If you don't want these as defaults, then new options have been provided to override the defaults. For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239.

- If a version of Fault Analyzer prior to V8.1 is used in parallel with V8.1, then one of these situations applies:
  - Do not use AUTO-managed history files (default for new PDSE history files with V8.1, or set using the IDIUTIL batch utility SetMinFaultEntries control statement).
  - Install the applicable compatibility PTF for the older version first, as per the following:

### Version

PTF

V7.1      UK30778

---

## Migrating from V6.1 to V7.1

This section provides information about changes to Fault Analyzer version 7.1 that you should be aware of if migrating from an earlier version.

- Prior to version 7, only the first available user exit specified using the Exits (see “Exits” on page 473) or DumpRegistrationExits (see “DumpRegistrationExits” on page 466) options would be invoked. Now, all exits specified are attempted invoked.

Review all specifications of Exits or DumpRegistrationExits options to ensure that correct processing occurs if all specified exits are invoked.

- The following user exit data area fields are no longer available:

- CTL.QUIET\_OPT and CTL\_QUIET\_MSGLIST

Use the Quiet option instead (for details, see “Quiet” on page 495).

- CTL.NODUP\_NORMAL\_HOURS

Either use the NoDup(Normal(...)) option (for details, see “NoDup” on page 482), or control the designation of duplicate faults using an End Processing user exit instead.

- CTL.MAXMINIDUMPPAGES\_OPT

Either use the MaxMinidumpPages option (for details, see “MaxMinidumpPages” on page 482), or control the writing of the minidump using an End Processing user exit instead.

- EPC.SUPPRESS\_SYSMDUMP

Use EPC.SUPPRESS\_DUMP instead.

- EPC.DUPLICATE\_FAULT\_ID

Use ENVFAULT\_ID instead.

- EPC.SUPPRESS\_IDIMSG

Use the Quiet option instead (for details, see “Quiet” on page 495).

- NFY.SUPPRESS\_IDIMSG

Use the Quiet option instead (for details, see “Quiet” on page 495).

Review all user exits for any usage of these fields.

- Normal duplicate detection has been enabled by default.

If normal duplicate detection is not already enabled using the NoDup(Normal(...)) option, and it is not desired, then it is necessary to add a NoDup(Normal(0)) option to the IDICNF00 parmlib member.

- The ability to suppress all information-level messages using the Quiet option without any suboptions specified has been removed, due to issues with messages being inadvertently suppressed.

If the Quiet option is currently specified without any suboptions, then it is necessary to explicitly specify any messages that should be suppressed, regardless of severity level.

- Minidumps in fault entries created with Fault Analyzer version 7 are not accessible to versions of Fault Analyzer prior to version 6. It might not be possible to reanalyze fault entries created with Fault Analyzer version 7 on versions prior to version 6.

- The DeferredReport option has been enabled for CICS by default.

If you did not previously specify the DeferredReport option, and the DeferredReport option should not be in effect for CICS, then it is necessary to override the default. One possible way to do this is by adding the following option to your IDICNF00 parmlib member, or an IDIOPTS user options file that is used by the CICS region:

NoDeferredReport

For details about changes to this option, see “DeferredReport” on page 463.

- The NoDup suboption of the RetainDump option is no longer supported.

## Migrating from V6.1 to V7.1

This suboption was replaced by the NoDup option (see “NoDup” on page 482) with APAR PQ53139 for Fault Analyzer version 2.1. Since then, specification of RetainDump(Auto,NoDup) was supported for backwards compatibility only. Specification of NoDup(Normal(24)) is the equivalent of the no longer supported RetainDump(Auto,NoDup) option.

- CICS users must ensure that the IDI.SIDIAUTH data set is added to the DFHRPL concatenation.  
Previously, IDI.SIDIMOD1 was required, but due to load modules having been moved, IDI.SIDIAUTH is now required instead.
- CICS users migrating from a version of Fault Analyzer prior to version 6.1 with APAR PK21990 (June 2006), must ensure that a shutdown PLT entry is added to their CICS regions. For details, see “Adding the required programs to the shutdown PLT” on page 303.

---

## LPA module compatibility

Fault Analyzer provides downward compatibility between load modules in the IDI.SIDIALPA data set. Hence, you can install a later version of Fault Analyzer, perform IPL with CLPA, and use these LPA modules with an earlier version of Fault Analyzer in LNKLST or STEPLIB<sup>3</sup>

**Note:** DB2 and IMS information might be incomplete in the analysis report, unless the earlier version is at the following or later PTF level:

V10      UK62342

V11 or later  
GA

---

## Sharing of history files across a sysplex with mixed levels of Fault Analyzer

There are no issues with sharing of history files across a sysplex with a mix of supported levels of Fault Analyzer installed.

---

3. All data sets in the STEPLIB concatenation must be APF-authorized in order for the Fault Analyzer data sets to retain their APF-authorization.

---

## Chapter 12. Preparing to customize Fault Analyzer

This section tells you how to customize Fault Analyzer for your particular installation, and how to set up global default options. There are times when you might want to set or change an option just for one job or reanalysis. Chapter 2, “Real-time analysis,” on page 15 and Chapter 3, “The Fault Analyzer ISPF interface,” on page 33 tells you how to adjust options in this case.

The global default options affect the way in which Fault Analyzer runs. For example, there are options to indicate what jobs should be analyzed, how much detail should be provided in the reports, and where compiler listings and side files can be located.

Before you can customize Fault Analyzer, you have to install it. SMP/E installation instructions are found in *Program Directory for IBM Fault Analyzer for z/OS*.

Installation-wide default options are contained in the parmlib member IDICNF00.

As part of the analysis process, Fault Analyzer attempts to find compiler listings or side files. Chapter 20, “Providing compiler listings or Fault Analyzer side files,” on page 285 suggests how to store listings or create and store side files, so that they are available to Fault Analyzer. This section also tells you which compiler options are required for IDILANGX processing.

It is a requirement for Fault Analyzer that REXX support is available via the standard MVS search path, if REXX user exits are called, or if diagnostic tracing is requested via the IDITRACE DDname.

The tasks described in the following assume that Fault Analyzer was installed into target libraries with a high-level qualifier of IDI. If you used a different high-level qualifier for your installation of Fault Analyzer, then substitute your high-level qualifier for IDI.

---

### Checklist for installing and customizing Fault Analyzer

In order to verify the installation of Fault Analyzer, and to start using Fault Analyzer at your site, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

**Note:** Copy all members of data set IDI.SIDISAM1 to another data set before proceeding, and make all changes to the copies only.

- 1. **Make Fault Analyzer modules available via LINKLIST and LPA**

For details, see “Making Fault Analyzer modules available” on page 231.

- 2. **Allocate a history file**

Although multiple history files might eventually be used at your site, a single history file is sufficient to verify the installation of Fault Analyzer.

There are no restrictions on the name of the history file, but the default name searched for by Fault Analyzer is IDI.HIST. If a different name is used, then the IDICNF00 parmlib member is used to provide the name through the DataSets option. You review the IDICNF00 parmlib member, and the options it might contain, later in the installation process.

## Checklist for installing and customizing Fault Analyzer

A suggested size of the initial history file is 100 cylinders.

General information about history files is provided in Chapter 18, “Setting up history files,” on page 261, along with considerations for choosing PDS or PDSE formats, and instructions for using the sample job provided for the data set allocation.

### — 3. Create the IDICNF00 parmlib member

For details, see Chapter 19, “Setting and changing default options for the site,” on page 279.

Ensure that a

```
DataSets(IDIHIST(ds))
```

option is included if you allocated a history file in step 2 on page 217 with a name other than IDI.HIST.

Likewise, if Fault Analyzer was installed using a high-level qualifier other than IDI, then the DataSets option must be used to provide the names of all required Fault Analyzer data sets.

### — 4. Define and initialize the message and abend code explanation repository

For details, see “Setting up the message and abend code explanation repository” on page 233.

### — 5. Install the MVS change options/suppress dump exit IDIXDCAP

For details, see “Installing the MVS change options/suppress dump exit IDIXDCAP” on page 253.

Information about the characteristics of this exit are provided in “Exits for invoking Fault Analyzer” on page 223.

At the completion of this step, Fault Analyzer is effectively enabled at your site, and might start analyzing abends and creating entries in your history file.

### — 6. Enable the Language Environment abnormal termination exit IDIXCEE

For details, see “Enabling the Language Environment abnormal termination exit IDIXCEE” on page 254.

For information to help you determine the applicability of this exit at your site, see “Exits for invoking Fault Analyzer” on page 223 and “Language Environment options required for invocation of Fault Analyzer” on page 228.

If this exit is not installed, then abends in LE-enabled programs are only captured if the IDIXDCAP exit is installed, and the LE TERMTHDACT option with any of the UA\* values (such as UATRACE or UADUMP) is in effect.

### — 7. Install the SVC dump registration exit IDIXTSEL

For details, see “Installing the MVS post-dump exit IDIXTSEL” on page 313.

This exit is required for:

- SVC recovery fault recording (RFR) dumps (see “Recovery fault recording” on page 30).
- Java analysis (see “Invoking Fault Analyzer from Java catch block” on page 25).
- CICS system dumps (see “Dump registration processing” on page 27)

### — 8. Install USERMOD IDITABD to eliminate the need for jobs to include an MVS dump DD statement

## Checklist for installing and customizing Fault Analyzer

- For details, see “Eliminating the need for a dump DD statement (++)IDITABD” on page 255.
- **9. Customize the CICS environment**

This step is only applicable if you are using CICS.  
For details, see Chapter 21, “Customizing the CICS environment,” on page 301.
  - **10. Customize the DB2 environment**

This step is only applicable if you are using DB2.  
For details, see Chapter 22, “Customizing the DB2 environment,” on page 317.

**Note:** It is recommended that the DB2 table index discussed in “Improving Fault Analyzer DB2 performance” on page 317 is created, as severe Fault Analyzer performance degradation might otherwise result when accessing DB2 catalog information.
  - **11. Customize the IMS environment**

This step is only applicable if you are using IMS.  
For details, see Chapter 23, “Customizing the IMS environment,” on page 319.
  - **12. Customize for ISPF**

For details, see Chapter 15, “Modifying your ISPF environment,” on page 247.
  - **13. Start the Fault Analyzer IDIS subsystem**

For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239.
  - **14. Add entry in IFAPRDxx parmlib member**

For details, see “Registering Fault Analyzer in the IFAPRDxx parmlib member” on page 238.

### Optional installation steps

- **1. Add BPX security server program control profile for Fault Analyzer programs**

This step is only required if program control has been activated for your installation.  
For details, see “Defining program control access to Fault Analyzer programs” on page 232.
- **2. Specify the name of the LE runtime library using an IDIOPTLM configuration-options module (optional)**

For details, see “Identifying the LE runtime library” on page 257.
- **3. Install USERMOD IDISPLI or IDISPLIA to enable implicit Fault Analyzer invocation from PL/I V2R3 applications (optional)**

For details, see “Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++)IDISPLI/++IDISPLIA” on page 255.
- **4. Change the default recovery fault recording IEATDUMP data set name using an IDIOPTLM configuration-options module (optional)**

For details, see “Changing the default recovery fault recording IEATDUMP data set name” on page 258.
- **5. Define XFACILIT resource classes to manage recovery fault recording data sets (optional)**

## **Checklist for installing and customizing Fault Analyzer**

- For details, see “Managing recovery fault recording data set access” on page 234.
- **6. Customize for Japanese language support**  
This step is only required if the Japanese feature of Fault Analyzer is installed.  
For details, see Chapter 24, “Customizing the Fault Analyzer Japanese feature,” on page 321.
  - **7. Customize for Korean language support**  
This step is only required if the Korean feature of Fault Analyzer is installed.  
For details, see Chapter 25, “Customizing the Fault Analyzer Korean feature,” on page 323.
  - **8. Install optional non-ISPF interfaces to access Fault Analyzer history files**  
For details, see Chapter 32, “Installing non-ISPF interfaces to access Fault Analyzer history files,” on page 443.
  - **9. Grant history file administrator authorization for change of settings via ISPF interface**  
For details, see “Restricting change of history file settings” on page 233.
  - **10. Review the chapter "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products" in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* for information about steps required to prepare your programs for use with IBM Problem Determination Tools products.**
  - **11. Set SLIP traps to capture documentation for selected Fault Analyzer error messages**  
For details, see “Setting Fault Analyzer SLIP traps” on page 229.

### Installation verification

- **1. Perform assembler IVP**  
For details, see “Verifying the use of Fault Analyzer with assembler” on page 325.
- **2. Perform COBOL IVP**  
Only perform this step if COBOL is installed at your site.  
For details, see “Verifying the use of Fault Analyzer with COBOL” on page 326.
- **3. Perform PL/I IVP**  
Only perform this step if PL/I is installed at your site.  
For details, see “Verifying the use of Fault Analyzer with PL/I” on page 327.
- **4. Perform IDIXCEE Language Environment exit IVP**  
For details, see “Verifying the IDIXCEE Language Environment exit enablement” on page 329.
- **5. Perform IDITABD USERMOD IVP**  
For details, see “Verifying the IDITABD USERMOD installation” on page 329.
- **6. Perform CICS IVP**  
Only perform this step if CICS is installed at your site.  
For details, see “Verifying the customization of Fault Analyzer under CICS” on page 329.
- **7. Perform DB2 IVP**

## Checklist for installing and customizing Fault Analyzer

Only perform this step if DB2 is installed at your site.

Both a C and a COBOL IVP is provided—for details, see “Verifying the use of Fault Analyzer with DB2” on page 331.

### 8. Perform ISPF IVP

For details, see “Verifying the use of Fault Analyzer through ISPF” on page 335.

Additional customization can optionally be performed using user exits as described in Chapter 31, “Customizing Fault Analyzer by using user exits,” on page 359. However, no user exits are required for Fault Analyzer to run.

## Library names after you finish installing

The following data sets should exist after you have completed the SMP/E APPLY of Fault Analyzer:

Data Set Name	Containing
IDI.SIDIALPA	Load modules that must be made available from LPA.
IDI.SIDIAUTH	Authorized load modules that must be made available from LINKLIST.
IDI.SIDIDOC1	Message and abend code explanation override files, sample reports, and Fault Analyzer User’s Guide and Reference PDF.
IDI.SIDIDOC2	Message and abend code explanation repository input data.
IDI.SIDIEXEC	The only purpose of this data set is to be used as input by the IDISVENU job described at “Setting up the message and abend code explanation repository” on page 233, which allocates and initializes the message and abend code repository.
IDI.SIDLPA1	REXX execs.
IDI.SIDIMAPS	Load modules that can be placed in LPA to minimize the space required by Fault Analyzer in the abending region when performing real-time analysis.
IDI.SIDIMLIB	Control block maps.
IDI.SIDIMOD1	ISPF message members.
IDI.SIDIPLIB	Non-authorized load modules that must be made available from LINKLIST.
IDI.SIDISAM1	ISPF panels.
IDI.SIDISLIB	Softcopy samples and installation jobs.
IDI.SIDITLIB	ISPF skeletons.
	ISPF tables.

### Japanese feature note

If the Japanese feature of Fault Analyzer is installed, the following extra data sets should exist:

Data Set Name	Containing
IDI.SIDIDJPN	Japanese message and abend code explanation override files, sample reports, and Fault Analyzer User’s Guide and Reference PDF.
IDI.SIDIMJPN	Japanese ISPF message members.
IDI.SIDIPJPN	Japanese ISPF panels.
IDI.SIDISJPN	Japanese ISPF skeletons.
IDI.SIDITJPN	Japanese ISPF tables.
IDI.SIDIXJPN	Japanese softcopy samples and installation jobs.

### End of Japanese feature note

### Korean feature note

If the Korean feature of Fault Analyzer is installed, the following extra data sets should exist:

Data Set Name	Containing
IDI.SIDIDKOR	Korean message and abend code explanation override files, sample reports, and Fault Analyzer User's Guide and Reference PDF.
IDI.SIDIMKOR	Korean ISPF message members.
IDI.SIDIPKOR	Korean ISPF panels.
IDI.SIDISKOR	Korean ISPF skeletons.
IDI.SIDITKOR	Korean ISPF tables.
IDI.SIDIXKOR	Korean softcopy samples and installation jobs.

### End of Korean feature note

## Storage recommendations

The real-time execution following an abend requires extra storage in the abending region while the analysis is carried out on the in-storage data.

The following are the requirements for the **minimum** available region size, assuming that neither Language Environment, nor Fault Analyzer, are available from LPA:

- A minimum of 440 kilobytes below-the-line (24-bit) storage, regardless of execution environment.
- A minimum of 32 megabytes above-the-line (31-bit) storage for CICS transactions.
- A minimum of 30 megabytes above-the-line (31-bit) storage for programs other than CICS transactions.

Depending on the type of fault being analyzed, and the environment in which this fault occurs, more storage might be required.

The storage requirements under CICS are for MVS GETMAIN-managed storage, not CICS DSA-managed storage. So, to increase below-the-line MVS GETMAIN-managed storage, you would need to decrease CICS below-the-line DSA-managed storage (and similarly for above-the-line storage).

Information about the actual storage used by Fault Analyzer is available at the end of the real-time analysis report. However, the amount of storage provided in the report accounts for the explicit allocations performed by Fault Analyzer only and does not include, for example, Language Environment heap and stack storage or storage used for load modules.

In post-abend situations, where the minidump and/or SYSDUMP is being processed, only a marginal increase in storage requirements occur over that of the real-time execution, as the result of allocating space for referenced dump pages. The increase is typically less than 500 kilobytes.

For interactive reanalysis, the storage is required in the TSO region.

The minimum available region size above-the-line can be reduced by the size of required modules that are either available from LPA (and therefore do not need to be loaded), or those that are already loaded, if, for example, the abending program uses LE.

Having LE in LPA saves around 7 megabytes, and Fault Analyzer in LPA around 12 megabytes, reducing the storage requirement for a typical non-CICS program to around 11 megabytes.

If the necessary below-the-line (24-bit) size is not available, then message IDI0086E is issued and processing terminates.

If the necessary above-the-line (31-bit) size is not available, then message IDI0055E is issued and processing terminates. Additionally, message IDI0087I might be issued to provide information about storage that could be made available if the command included in the message text is issued to add modules to LPA. The module names likely to be included in the message are the Fault Analyzer modules IDIDA and IDILANGX. To place these modules in LPA, and save approximately 12 megabytes above-the-line (31-bit) storage, either issue the following MVS operator command:

```
SETPROG LPA,ADD,MOD=(IDIDA, IDILANGX),DSN=LNKLST
```

or add IDI.SIDILPA1 and IPV.SIPVLPA1<sup>4</sup> to the LPALSTxx parmlib member (for details, see "Making Fault Analyzer modules available" on page 231).

**Note:** If Fault Analyzer modules are loaded into LPA, then it is important that step 2b1 on page 353 is performed after apply of any Fault Analyzer maintenance.

Failure to perform this step following the installation of maintenance prevents the update of Fault Analyzer LPA modules. Because all Fault Analyzer modules are not in LPA, the result can be a mismatch between the old and the new code, which might lead to undefined behavior.

The MVS IEFUSI exit can be used as a general way to provide a larger region size if JCL change is not practical for all jobs. A sample IEFUSI exit is provided as member IDISUSI in the IDI.SIDISAM1 data set. The exit increases the region size of all jobs by 16 megabytes. Refer to the comments in the sample for details about how to install the exit.

---

## Exits for invoking Fault Analyzer

There are a number of exits provided with Fault Analyzer to invoke it for real-time analysis of an abend, or for SVC dump registration. All must be installed to ensure that Fault Analyzer is invoked for all applicable abend situations.

Because CICS has a unique transaction dispatching mechanism, the invocation exits for CICS are unique. The non-CICS execution environments are generally referred to as "batch", meaning anything which is not CICS, including for example IMS.

## Invocation for non-CICS transaction abends

The following exits all invoke Fault Analyzer for real-time analysis when an abend, other than a CICS transaction abend, occurs (for example, batch and IMS).

---

4. Data set created as part of the IBM Problem Determination Tools for z/OS Common Component (PDTCC) installation.

## Exits for invoking Fault Analyzer

### MVS IEAVTABX change options/suppress dump exit IDIXDCAP

Characteristics:

- This exit can be used with both Language Environment and non-Language Environment based batch application programs.
- Installation depends on the level of z/OS, as follows:

#### For z/OS 2.2 and later levels of z/OS

IDIXDCAP is installed as an IEAVTABX\_EXIT dynamic exit and is invoked for all abends, regardless of whether the job step has allocated a SYSMDUMP, SYSUDUMP, or SYSABEND DDname.

#### For levels of z/OS prior to z/OS 2.2

IDIXDCAP is installed in the MVS IEAVTABX CSECT exit list.

The MVS IEAVTABX exit process is only called by MVS if the job step has allocated a SYSMDUMP, SYSUDUMP, or SYSABEND DDname, or if the IDITABD USERMOD has been applied.

**Note:** A matching SLIP TRAP with ACTION=NODUMP<sup>5</sup> prevents the MVS IEAVTABX exit from being called (for example, a CANCEL command resulting in an Sx22 abend for which most MVS systems have a matching SLIP TRAP). To facilitate the Fault Analyzer analysis, either disable the matching SLIP TRAP or, for LE-based applications, use the batch LE abnormal termination CEEEXTAN CSECT exit IDIXCEE instead (see below).

- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.
- LE-enabled abends need to run with TERMTHDACT, specifying the suboption UATRACE, UADUMP, UAONLY, or UAIMM, in the LE options so that LE calls a system dump to activate this exit. All other TERMTHDACT suboption settings skip the IEAVTABX exit invocation and invoke the CEEEXTAN exit (described below) instead.
- This exit is APF authorized and therefore able to extract job related messages from the console.

For information about installation of this exit, refer to “Installing the MVS change options/suppress dump exit IDIXDCAP” on page 253.

### Batch LE abnormal termination CEEEXTAN CSECT exit IDIXCEE

Characteristics:

- This exit is only effective with Language Environment based batch application programs.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.
- If the LE option TERMTHDACT is used with the UATRACE, UADUMP, UAONLY, or UAIMM suboption, then the MVS change options/suppress dump exit is invoked instead of the LE abnormal termination exit.
- Permits instance-specific LE message inserts to be obtained, and thus included in the analysis report.

5. The use of ACTION=NOSVCD or ACTION=(NOSYSA,NOSYSM,NOSYSU) is not sufficient to ensure that Fault Analyzer is not invoked through the IDIXDCAP exit.

For information about installation of this exit, refer to “Enabling the Language Environment abnormal termination exit IDIXCEE” on page 254.

**Note:** This exit is easily confused with LE's CICS abnormal termination exit, CEECXTAN (Fault Analyzer exit IDIXCCEE). Pay special attention if both this exit and the CICS exits are installed.

If both the batch Language Environment abnormal termination exit IDIXCEE and the MVS change options/suppress dump exit IDIXDCAP are installed, then the IDIXDCAP exit intercepts the abend instead of the LE exit if one of the following LE options is in effect:

- TERMTHDACT(UATRACE)
- TERMTHDACT(UADUMP)
- TERMTHDACT(UAONLY)
- TERMTHDACT(UAIMM)

and a SYSABEND, SYSUDUMP, or SYSMDUMP DDname is allocated (or the IDITABD USERMOD has been applied).

### Summary of exit usage

Figure 128 on page 226 shows the exit used to invoke Fault Analyzer, depending on execution environment, options in effect, specification of MVS dump DD statement, and the IDITABD USERMOD.

## Exits for invoking Fault Analyzer

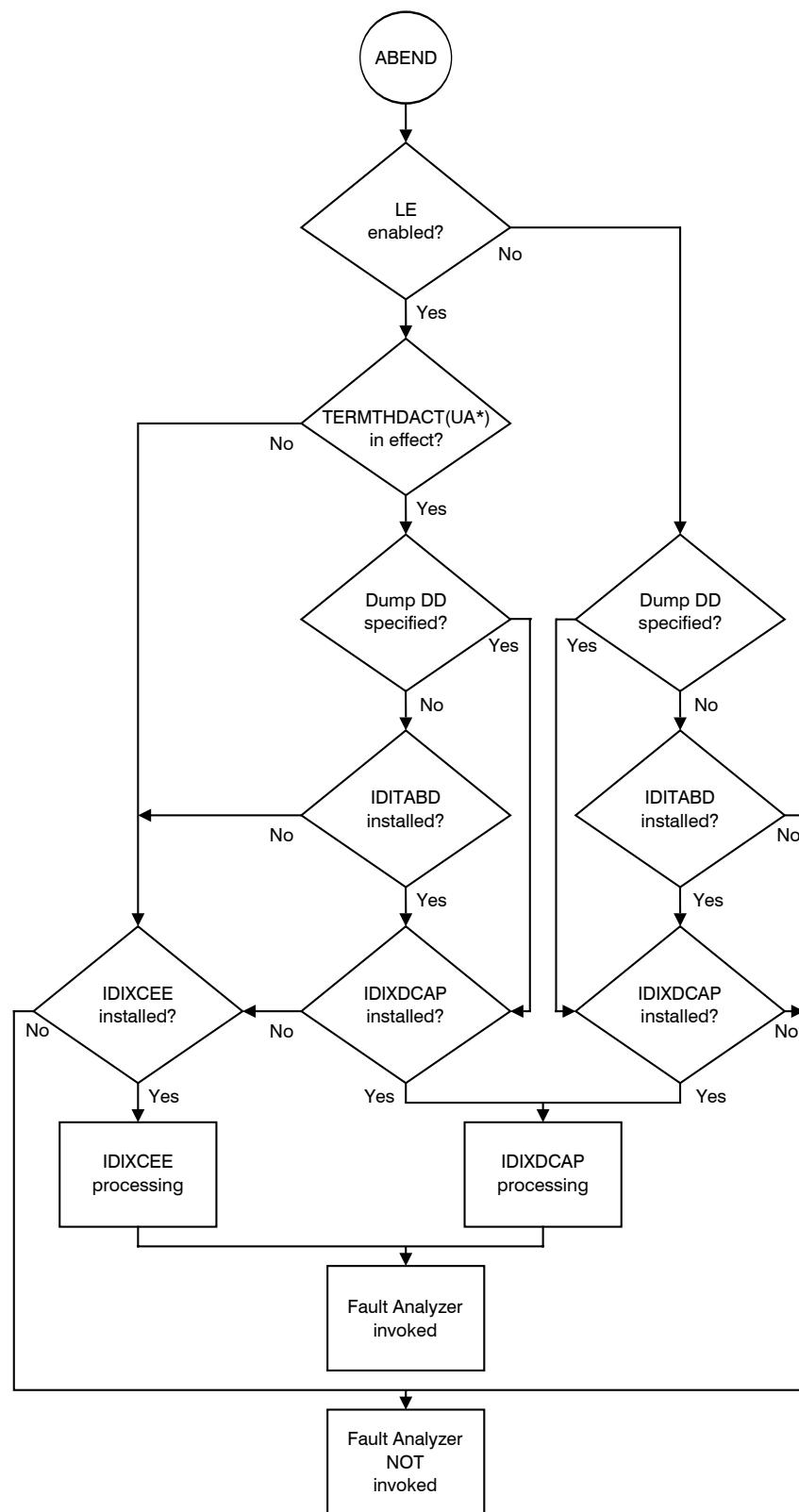


Figure 128. Summary of Fault Analyzer non-CICS (batch) invocation exit usage

## Invocation for CICS transaction abends

The following exits all invoke Fault Analyzer for real-time analysis when a CICS transaction abend occurs.

### CICS XPCABND and XDUREQ global user exit IDIXCX52 or IDIXCX53

Characteristics:

- This exit is provided to invoke Fault Analyzer for CICS transaction abend analysis.
- All transaction abends can be captured using this exit, except for U1xxx or U4xxx-type abends in Language Environment based applications. These transaction abend types can be handled by also installing the CICS LE abnormal termination CEECXTAN CSECT exit, IDIXCCEE, described below.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
  - Reanalysis of faults that were captured using this exit can be performed if a minidump was written.

For information about installation of this exit, refer to Chapter 21, "Customizing the CICS environment," on page 301.

### CICS LE abnormal termination CEECXTAN CSECT exit IDIXCCEE

Characteristics:

- This exit is only effective with Language Environment based CICS application programs.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.
- The LE option TERMTHDACT does not affect the invocation of this exit.

For information about installation of this exit, refer to "Configuring Language Environment for CICS to invoke Fault Analyzer" on page 302.

## SVC dump registration

An exit is provided with Fault Analyzer for SVC dump registration into a history file.

### MVS post-dump IEAVTSEL CSECT exit IDIXTSEL

Characteristics:

- This exit is invoked whenever an SVC dump is written by the DUMPSRV address space.
- The use of this exit requires the Fault Analyzer IDIS subsystem to be active—for information on this, see Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239.
- No analysis is performed, but a dump registration fault entry is created. When this fault entry is first reanalyzed, then a report and minidump is added.
- This exit is primarily intended for recording of CICS system dumps and recovery fault recording SDUMPs.

For information about installation of this exit, refer to "Installing the MVS post-dump exit IDIXTSEL" on page 313.

---

## **Language Environment options required for invocation of Fault Analyzer**

The need for specific LE options to capture real-time abends depends on the execution environment, as described in the following.

### **LE options required for non-CICS abends**

If the CEEEXTAN LE abnormal termination exit (IDIXCEE) has been installed, then there are no specific LE options required for Fault Analyzer to be invoked for an abend. If, however, the MVS IEAVTABX change options/suppress dump exit (IDIXDCAP) is to be used to capture LE abends, then an option to cause LE to take a SYSABEND, SYSUDUMP, or SYSMDUMP, such as TERMTHDACT(UADUMP), is required. This option permits the IEAVTABX exit to gain control when the MVS dump is about to be written.

**Note:** If both the CEEEXTAN and IEAVTABX exits are installed, then Fault Analyzer resolves the processing to perform only one analysis of a fault.

### **LE options required to capture Java application abends**

In order to capture abends in a Java application with Fault Analyzer, the LE TERMTHDACT option must be in effect with one of the UA\* suboptions, for example TERMTHDACT(UAIMM).

This option can be set by issuing the command

```
oedit .profile
```

from an OMVS session to edit the user profile, and then specifying

```
export _CEE_RUNOPTS="TERMTHDACT(UAIMM)"
```

### **LE options required for CICS abends**

There are no required LE options applicable to CICS abends.

For CICS trace considerations, see also “Preventing LE from causing the CICS trace to wrap” on page 312.

---

## **Running Fault Analyzer with similar third-party products**

In general, Fault Analyzer works with other similar third-party products without problems, with the possible exception being Language Environment batch jobs. Fault Analyzer uses the Language Environment CEEEXTAN facility (IDIXCEE), as does potentially other similar third-party products. If more than one exit is specified in the CEEEXTAN list, then Fault Analyzer should be the first exit specified.

The analysis of LE jobs by Fault Analyzer can also be done using the IEAVTABX MVS change options/suppress dump exit (IDIXDCAP). Ensure that the LE options include the TERMTHRDACT option, with the UATRACE, UADUMP, UAONLY, or UAIMM suboption, to make LE request an MVS dump for an abend. This way, the third-party product can use the CEEEXTAN exit while Fault Analyzer can run from the IEAVTABX MVS change options/suppress dump exit. The Fault Analyzer IDIXDCAP exit analyzes LE abends exactly the same way as the IDIXCEE exit. This similarity is because the exits do not actually do the analysis, they simply provide the method of invoking Fault Analyzer. It is the same Fault Analyzer analysis engine that runs for all Fault Analyzer exits, including CICS.

## Running Fault Analyzer with similar third-party products

For non-LE batch there are no known conflicts between Fault Analyzer and similar third-party products. However, it is recommended to specify the RETAINDUMP(ALL) option in the IDICNF00 parmlib member to ensure that similar third-party products that might rely on the MVS dump being taken for their invocation are not affected. Once Fault Analyzer is the only abend analysis product installed, then the RETAINDUMP(ALL) option can be removed.

Users of the CICS pre-transaction dump global user exit (XDUREQ) might be affected by Fault Analyzer's use of the pre-abend (XPCABND) exit. Specifically, Fault Analyzer's default action after invocation via the XPCABND exit is to suppress transaction dumps. If suppressed, CICS does not then subsequently invoke any XDUREQ exit programs. Users who require the XDUREQ exit to be invoked by CICS should use the Fault Analyzer RetainCICSdump(ALL) option.

---

## MVS dump data set size

With Fault Analyzer installed, you should expect an increase in the size of any MVS system dump taken. It might be necessary to review your dump data set allocation size parameters.

---

## Application-handled error conditions

You can write application error handlers to completely suppress any indications of an abend, or on their completion, allow abend processing to continue.

Generally, abend processing is allowed to continue after the error handler has completed its task. However, if error handlers for application programs are not letting normal abend termination occur, and you want to invoke Fault Analyzer for such applications, then it might be necessary to either disable the application error handling, or add a call to IDISNAP (for details, see "Using the program SNAP interface (IDISNAP)" on page 20).

An example of an application error handling routine that does not invoke Fault Analyzer is a PL/I "ON ERROR" block, that either calls PLIDUMP with the 'S' option, or issues STOP.

A PL/I USERMOD is available to always invoke Fault Analyzer when calling PLIDUMP, even if using the 'S' option—for details, see "Always invoking Fault Analyzer from PL/I PLIDUMP (++IDISPDM)" on page 255.

---

## Setting Fault Analyzer SLIP traps

As documentation for certain situations where Fault Analyzer issues an error message, an MVS dump is often required by your IBM service representative.

In most cases, Fault Analyzer automatically writes a Recovery Fault Recording (RFR) dump, which serves two purposes:

- To enable reanalysis of the RFR fault entry, effectively making it transparent to the user that an error has occurred.
- To provide your IBM service representative with information about the error, if suspected of being a Fault Analyzer defect.

If such RFR dumps are being written (which requires the Fault Analyzer IDIS subsystem to be started and the IDIXTSEL exit installed - see "Recovery fault recording" on page 30 for more information), then there is generally no need to set SLIP traps for message IDs other than those marked with '\*' below.

## Setting Fault Analyzer SLIP traps

The following shows a sample SLIP trap to capture an SVC dump in case a particular Fault Analyzer message is issued:

```
SL SET, ID=xxxx,MSGID=zzzzzzzz,ACTION=SVCD,END
```

where

xxxx is a unique SLIP trap identifier (for example, F047)

zzzzzzzz is the message ID.

Refer to "MVS System Commands" for the complete syntax of the SLIP command and for additional parameters that might be considered, such as MATCHLIM.

SLIP traps for the following message IDs should be considered:

IDI0047S

IDI0092S

IDI0105S

IDI0123S \*

IDI0144E \*

IDI0168E \*

\* These message IDs are not subject to Recovery Fault Recording (RFR) processing and MVS dumps are therefore never written automatically by Fault Analyzer for these.

---

# Chapter 13. Customizing the operating environment for Fault Analyzer

This chapter provides information about customization of the operating environment required to run Fault Analyzer.

## Making Fault Analyzer modules available

The following must be performed in order to make Fault Analyzer modules available.

### 1. Authorizing IDI.SIDIAUTH and adding to the LINKLIST

Fault Analyzer modules that can reside in a PDS and require APF-authorization are placed in target library, IDI.SIDIAUTH. You **must** APF-authorize IDI.SIDIAUTH by adding it to the IEAAPFx or PROGxx member (if available on your system) in SYS1.PARMLIB. IDI.SIDIAUTH must also reside in the LINKLIST. Add IDI.SIDIAUTH to your concatenated LINKLIST via the LNKLSTxx or PROGxx member in your SYS1.PARMLIB.

**Note:** MVS requires that data sets in LINKLIST either be in the master catalog or specified with the volume serial number where the data set resides.

The following load modules in IDI.SIDIAUTH do not execute correctly unless they are loaded from an APF-authorized library:

IDIDA  
IDIPDDIR  
IDIUTIL

### 2. Authorizing IDI.SIDIAUT2 and adding to the LINKLIST

Fault Analyzer modules that must reside in a PDSE and require APF-authorization are placed in the target library, IDI.SIDIAUT2. You **must** APF-authorize IDI.SIDIAUT2 by adding it to the IEAAPFx or PROGxx member (if available on your system) in SYS1.PARMLIB. IDI.SIDIAUT2 must also reside in the LINKLIST. Add IDI.SIDIAUT2 to your concatenated LINKLIST via the LNKLSTxx or PROGxx member in your SYS1.PARMLIB.

**Note:** MVS requires that data sets in LINKLIST either be in the master catalog or specified with the volume serial number where the data set resides.

### 3. Adding IDI.SIDIMOD1 and IPV.SIPVMODA<sup>6</sup> to the LINKLIST

To enable Fault Analyzer to work correctly, you **must** add IDI.SIDIMOD1 to your concatenated LINKLIST. To do this, add this library to either your LNKLSTxx or PROGxx (if available on your system) member in SYS1.PARMLIB.

Data set IPV.SIPVMODA must be added to LINKLIST in order for Fault Analyzer to provide source level support when using compiler listings or SYSADATA files.

**Note:** MVS requires that data sets in LINKLIST either be in the master catalog or specified with the volume serial number where the data set resides.

The IBM Problem Determination Tools for z/OS Common Component also provides a data set IPV.SIPVMOD1, which is not required by Fault Analyzer.

---

6. Data set created as part of the IBM Problem Determination Tools for z/OS Common Component (PDTCC) installation.

## Making Fault Analyzer modules available

### 4. Adding IDI.SIDIALPA to the LPALIST

Fault Analyzer modules that must be loaded into the LPA reside in the target library, IDI.SIDIALPA. Add IDI.SIDIALPA to your concatenated LPALIST via the LPALSTxx member in your SYS1.PARMLIB.

**Note:** MVS requires that data sets in LPALIST are either in the master catalog or specified with the volume serial number where the data set resides. The LPALSTxx change must then be implemented by performing an IPL with CLPA.

### 5. Adding IDI.SIDLPA1 and IPV.SIPVLPA1<sup>7</sup> to the LPALIST (optional)

Fault Analyzer modules that can optionally be loaded into the LPA reside in the target library, IDI.SIDLPA1, while equivalent IBM Problem Determination Tools for z/OS Common Component (PDTCC) modules used by Fault Analyzer reside in library IPV.SIPVLPA1. If these libraries are added to your LPALIST, then less space is required for Fault Analyzer in the abending region being analyzed. To conserve the maximum amount of virtual storage, add IDI.SIDLPA1 and IPV.SIPVLPA1 to your concatenated LPALIST via the LPALSTxx member in your SYS1.PARMLIB. All modules in IDI.SIDLPA1 and IPV.SIPVLPA1 are 31-bit and are therefore automatically loaded into ELPA above the line.

**Note:**

- a. MVS requires that data sets in LPALIST are either in the master catalog or specified with the volume serial number where the data set resides. The LPALSTxx change must then be implemented by performing an IPL with CLPA.
- b. If the IDI.SIDLPA1 data set is added to LPALIST, then care must be taken when applying maintenance to Fault Analyzer. For details, see Chapter 29, "Maintaining Fault Analyzer," on page 353.

### 6. Performing IPL with CLPA or running IDICZSVC

It is necessary to IPL your system again, with CLPA, as the Fault Analyzer installation has added SVC modules to LPA in data set IDI.SIDIALPA. Failure to do this IPL results in abend S16D being issued when Fault Analyzer performs analysis.

For the initial install, or any time later when module IDICSVCR in IDI.SIDIALPA is updated, you can do the following if an IPL cannot be scheduled:

- a. Issue the operator command:

```
SETPROG LPA,ADD,MOD=(IDICSVCR),DSN=IDI.SIDIALPA
```

- b. Submit a batch job containing the following EXEC JCL statement:

```
// EXEC PGM=IDICZSVC
```

This statement dynamically installs the Fault Analyzer SVC 109 ESR code 53 to your system.

---

## Defining program control access to Fault Analyzer programs

If security server program control is activated for your installation, for example due to z/OS Unix System Services (BPX) server requirements, then a PROGRAM class profile that identifies all Fault Analyzer programs as being controlled programs can be defined using the command:

---

7. Data set created as part of the IBM Problem Determination Tools for z/OS Common Component (PDTCC) installation.

## Defining program control access to Fault Analyzer programs

```
RDEFINE PROGRAM IDI* ADDMEM('IDI.SIDIAUTH'//NOPADCHK) UACC(READ)
```

(Refer to your security server's documentation for details.)

Failure to define Fault Analyzer programs as being controlled might result in messages, such as CSV042I, ICH420I, ICH422I, or BPX014I, being issued, if an abend occurs in a z/OS Unix System Services server region.

## Restricting change of history file settings

By default, all users with UPDATE access to a history file can change the history file prefix, or the maximum/minimum number of fault entries, using either of the following methods:

- The Fault Analyzer ISPF interface action-bar option "File->Change Fault History File Settings..." (for details, see "Change fault history file settings" on page 57).
- The IDIUTIL batch utility SetFaultPrefix, SetMaxFaultEntries and SetMinFaultEntries functions (for details, see "IDIUTIL control statements" on page 338).

To restrict the change of settings for a given history file using either of the above methods, the security administrator can define an IDI\_ADMIN XFACILIT profile for the history file, to which access can be granted as appropriate.

### Syntax

```
►►IDI_ADMIN.history-file-dsn►►
```

where *history-file-dsn* is the fully qualified data set name of the history file.

To enable changing history file settings once the IDI\_ADMIN XFACILIT profile has been defined, a user must have both of the following access permissions:

- UPDATE (or greater) access to the IDI\_ADMIN XFACILIT profile
- UPDATE (or greater) access to the history file, either via normal security server data set profiles, or via XFACILIT (for details, see "Managing history file fault entry access" on page 273).

The following are sample RACF commands to define an IDI\_ADMIN XFACILIT profile for history file MY.HIST and granting Fault Analyzer administrator authorization to change settings only for users who are members of group PAYROLL:

```
RDEFINE XFACILIT IDI_ADMIN.MY.HIST UACC(NONE)
PERMIT IDI_ADMIN.MY.HIST CLASS(XFACILIT) ID(PAYROLL) ACCESS(UPDATE)
```

## Setting up the message and abend code explanation repository

To enable Fault Analyzer message and abend code explanations, a VSAM cluster must be defined and initialized. This definition and initialization is done by submitting the sample job IDISVENU in the IDI.SIDISAM1 data set. If a name, other than the default name of IDI.IDIVSENU is used, then it is necessary to specify the DataSets option with the IDIVSENU suboption containing the name to be used. For details, see "Detail" on page 465.

## Setting up the message and abend code explanation repository

In situations where different versions of z/OS are rolled out across an installation, with different versions of Fault Analyzer, it might be convenient to specify the IDIVSENU data set using the &SYSR1 substitution symbol, for example:

```
DataSets(IDIVSENU(IDI.IDIVSENU.&SYSR1.))
```

and then allocate these VSAM data sets with the SYSRES VOLSER on the respective SYSRES volumes. That way, the IDICNF00 parmlib member DataSets specification of IDIVSENU is always correct, no matter which SYSRES volume is being IPL'ed from.

Ensure that all users have READ access to the data sets defaulted to, or specified using, the IDIDOC and IDIVSENU DDnames. Additionally, due to the occasional automatic updating of this file by the IDIS subsystem with corrections or new explanations, the IDIS subsystem should be granted UPDATE access to the IDIVSENU VSAM KSDS data set.

### Japanese feature note

For the Japanese feature of Fault Analyzer, an extra repository must be set up using sample job IDISVJPN in the IDI.SIDIXJPN data set. If a name, other than the default name of IDI.IDIVSJPN is used, then it is necessary to use the IDIVSJPN suboption of the DataSets option to specify the name to be used. For details, see “Detail” on page 465.

Ensure that all users have READ access to the data set defaulted to, or specified using, the IDIVSJPN DDname. Additionally, due to the occasional automatic updating of this file by the IDIS subsystem with corrections or new explanations, the IDIS subsystem should be granted UPDATE access to the IDIVSJPN VSAM KSDS data set.

### End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, an extra repository must be set up using sample job IDISKOR in the IDI.SIDIXKOR data set. If a name, other than the default name of IDI.IDIVSKOR is used, then it is necessary to use the IDIVSKOR suboption of the DataSets option to specify the name to be used. For details, see “Detail” on page 465.

Ensure that all users have READ access to the data set defaulted to, or specified using, the IDIVSKOR DDname. Additionally, due to the occasional automatic updating of this file by the IDIS subsystem with corrections or new explanations, the IDIS subsystem should be granted UPDATE access to the IDIVSKOR VSAM KSDS data set.

### End of Korean feature note

## Managing recovery fault recording data set access

If Fault Analyzer's normal process of recording an application abend fails for exception conditions, such as insufficient virtual storage or an abend in Fault Analyzer, it then attempts to capture the abend situation with an MVS SDUMP (SVC dump) or MVS IEATDUMP (transaction dump or TDUMP). This process creates a separate dump data set, which is linked with the recovery fault recording history file fault entry. With this recovery process, Fault Analyzer is normally able

to provide interactive reanalysis of the initial application abend in spite of the exception condition Fault Analyzer encountered during the capture. The linked SDUMP or TDUMP provides the storage data that would normally have been recorded in the 'minidump' section of the fault entry if the capture had not had the exception.

Fault Analyzer controls the use of RFR SDUMP or TDUMP with XFACILIT security profiles. If the access to the SDUMP or TDUMP XFACILIT security profiles are not available or not defined, then no security violations are generated. This lack of violations is because Fault Analyzer checks the required user access first, and if not available does not issue the associated SDUMP or TDUMP request, although TDUMP is still used if the user has ALTER access to the TDUMP data set profile but no XFACILIT access.

Fault Analyzer tries to use SDUMP as the preferred dump type, and only if the necessary SDUMP access authorization is not available for the abending user ID, is the TDUMP access authorization checked. The dump process used by SDUMP is faster than the TDUMP process.

If the Fault Analyzer XFACILIT process described here is used as the SDUMP or TDUMP control of its RFR dumps, then the actual SDUMP or TDUMP can not be read or deleted by a normal end user, except through analysis or deletion of the fault entry it is linked to. For example, in a case where the 'payroll' application might have its own history file that general users don't have read access to, then this XFACILIT process means that any RFR SDUMPS or TDUMPS for 'payroll' are restricted from general users because they can't access the fault entries.

The XFACILIT access requirements for RFR SDUMP or TDUMP differ as explained in the following.

### SDUMP recovery fault recording data sets

When an SDUMP is requested, it is generated by the DUMPSRV address space with a naming convention that is determined by DUMPSRV. Normally most users on the system, except for the systems programmers, are restricted by UACC(NONE) to these SDUMPS. To be able to request an SDUMP, Fault Analyzer must use authorized state, which it obtains from its internal SVC process.

Fault Analyzer only uses SDUMP in the recovery fault recording process if the user ID under which the abend occurred was granted XFACILIT access using the following XFACILIT resource class setup.

### Using the XFACILIT resource class for SDUMP RFR data sets

To have Fault Analyzer use SDUMP in its RFR process, set up an XFACILIT class profile with the name IDI\_SDUMP\_ACCESS and provide ALTER access to the user IDs or groups where SDUMPS are required for RFR exceptions. The following define would permit Fault Analyzer to create SDUMPS for all users in the CICS group, if their fault entry create encounters an exception.

```
RDEF XFACILIT IDI_SDUMP_ACCESS UACC(NONE)
PERMIT IDI_SDUMP_ACCESS CLASS(XFACILIT) ID(CICS) ACCESS(ALTER)
```

The ALTER access is to the XFACILIT IDI\_SDUMP\_ACCESS profile, it is not to the actual SDUMP data sets. Fault Analyzer uses authorized state to permit access to RFR SDUMPS. The IDI\_SDUMP\_ACCESS profile acts as a switch Fault Analyzer can check to see if SDUMPS should be created for that user ID.

## Managing recovery fault recording data set access

If by chance a fault entry creation has an exception requiring an RFR dump, then Fault Analyzer only creates and links an SDUMP to the fault entry if the user has ALTER access to the XFACILIT IDI\_SDUMP\_ACCESS profile.

If a user doing problem analysis has read or delete access to a fault entry, and the fault entry has an SDUMP linked to it (the fault entry was created by a recovery fault recording exception), then Fault Analyzer provides the equivalent access to the SDUMP as an extension to the fault entry. Deleting a fault entry implicitly causes any linked SDUMP to be deleted.

Because capturing an SDUMP is usually much faster than capturing a TDUMP, then it is recommended that at least performance critical systems, such as CICS, are given authority to use RFR SDUMPs by granting the above access.

## TDUMP recovery fault recording data sets

This section describes the required authorization to create, read, and delete exception condition RFR TDUMPs. Because a TDUMP requester can nominate the TDUMP data set name, this section also describes the naming convention process.

**Note:** The older RFR TDUMP naming convention of user ID high-level qualifier, is no longer used because of the security and data set deletion problems that were frequently encountered with user ID TDUMP high-level qualifiers.

To overcome the security concerns of normal data set profiles with respect to TDUMP recovery fault recording data sets, Fault Analyzer supports the use of the XFACILIT resource class as described in the following. Together with the use of the XFACILIT resource class, it is recommended that UACC(NONE) is used as the general data set profile access level for TDUMP recovery fault recording data sets, to prevent the possibility of security exposures. The exposure would exist if ALTER access was granted to all users on the RFR TDUMP data set profile to permit creation, instead of UACC(NONE) and the XFACILIT set up.

If a system has a situation where all end users have similar access privileges, then the RFR TDUMPs are still taken if you choose to not set up the XFACILIT IDIRFR\_TDUMP\_HLQ, and instead give all users ALTER access to the TDUMP data set profile. This environment would probably have all users with equal access to the history files on that system. However, if some users do not have read access to all history files, then IDIRFR\_TDUMP\_HLQ and UACC(NONE) on the data set profile should be considered to extend the protection to any linked RFR TDUMPS.

## Using the XFACILIT resource class for TDUMP RFR data sets

To set up the XFACILIT resource class for Fault Analyzer TDUMP recovery fault recording data sets, the high-level qualifier must initially be determined.

The names of the recovery fault recording data sets created are determined by the IDIRFRDS CSECT. Fault Analyzer provides the default name prefix IDIRFRHQ.IDIRFR.\*, which can be changed by installing the IDISRFR USERMOD—for details, see “Changing the default recovery fault recording IEATDUMP data set name” on page 258.

**Note:** Although the term "qualifier" is used in singular throughout this section, one or more qualifiers can be used in the access control setup.

Given the high-level qualifier used, set up an XFACILIT class profile with the name

IDIRFR\_TDUMP\_HLQ.*hlq*.\*\*

where *hlq* is the recovery fault recording data set high-level qualifier.

If the high-level qualifier includes a symbol name such as TDUMP&SYSCLOSE., then it might be necessary to set up more than one profile, depending on the expected symbol substitution values.

Having defined the XFACILIT profile (or profiles, if more than one due to symbol substitution), then provide the appropriate level ALTER or NONE for the users concerned. If the user's access level to the XFACILIT class is ALTER, then through Fault Analyzer, the user implicitly has TDUMP create capability to the data set whose high-level qualifier, after any symbol substitution, matches the XFACILIT profile name *hlq* value.

General access of ALTER to an XFACILIT profile does not override any normal data set profile protecting a recovery fault recording data set. It only permits the necessary access authorization to the linked TDUMP data set when performing actions through Fault Analyzer, such as reading it during reanalysis, or deleting it when the associated fault entry is deleted.

Sometimes, by chance, a fault entry creation has an exception requiring an RFR dump. Then Fault Analyzer only creates and links a TDUMP to the fault entry under one of these conditions:

- You have ALTER access to the appropriate XFACILIT IDIRFR\_TDUMP\_HLQ profile.
- You have ALTER access to the TDUMP data set profile.

Fault Analyzer provides the equivalent access to the TDUMP as an extension to the fault entry when these two conditions apply:

- If you are doing problem analysis and have read or delete access to a fault entry.
- If the fault entry has a TDUMP linked to it (the fault entry was created by a recovery fault recording exception).

Deleting a fault entry implicitly causes any linked TDUMP to be deleted.

### RFR TDUMP XFACILIT example

The following is an example of the recommended set-up of the XFACILIT class for the purpose of managing the Fault Analyzer recovery fault recording data sets. This example can be modified or expanded on by an installation as required.

**Note:** If the IDISRFR USERMOD is installed to change the default high-level qualifier, then replace IDIRFRHQ by the actual high-level qualifier in the following examples.

1. Define an XFACILIT profile with the name IDIRFR\_TDUMP\_HLQ.IDIRFRHQ.\*\* and grant universal access of ALTER to this profile:  

```
RDEFINE XFACILIT IDIRFR_TDUMP_HLQ.IDIRFRHQ.** UACC(ALTER)
```
2. Define a generic data set profile for IDIRFRHQ.\* with universal access of NONE:  

```
ADDSD 'IDIRFRHQ.*' UACC(NONE)
```

---

## **Registering Fault Analyzer in the IFAPRDxx parmlib member**

If you purchased Fault Analyzer as part of product code 5655-PDS, IBM Problem Determination Solution Pack for z/OS, and have not already done so for another tool in product code 5655-PDS, then include an entry in the IFAPRDxx parmlib member as follows:

```
PRODUCT OWNER('IBM CORP')
           NAME('IBM PD SOLTN PAC')
           ID(5655-PDS)
           VERSION(*) RELEASE(*) MOD(*)
           FEATURENAME('PROB-DET-SOL-PAC')
           STATE(ENABLED)
```

Otherwise, if you have not purchased Fault Analyzer as part of product code 5655-PDS, then an entry in IFAPRDxx parmlib is not required. However, if you would prefer to have an entry, then the following can be added:

```
PRODUCT OWNER('IBM CORP')
           NAME('FAULT ANALYZER')
           ID(5655-Q11)
           VERSION(*) RELEASE(*) MOD(*)
           FEATURENAME('FAULT ANALYZER')
           STATE(ENABLED)
```

All parameters except VERSION, RELEASE and MOD must be specified exactly as shown. Any syntactically valid values can be specified for VERSION, RELEASE and MOD.

Refer to *MVS Initialization and Tuning Reference* for general information about the IFAPRDxx parmlib member.

## **Generic product enablement**

Fault Analyzer checks product enablement codes for the following values, and in the order listed:

### **5655-PDS**

IBM Problem Determination Solution Pack

### **5655-DMS**

IBM Problem Determination Modernization Solution Pack

### **5697-CDT**

IBM COBOL DevOps Tools Suite for z/OS V1.1

### **5655-Q11**

IBM Fault Analyzer

Sites that use a generic product enablement policy, for example,  
PRODUCT(\*) STATE(ENABLED)

might need to code extra statements to ensure that the correct product code has been enabled:

- If your product code is 5655-PDS, then no extra statements are required.
- If your product code is 5655-DMS or 5697-CDT, then code an entry for 5655-PDS, as per the above example, with STATE(DISABLED).
- If your product code is 5655-Q11, then code entries for both 5655-PDS, 5655-DMS and 5697-CDT with STATE(DISABLED).

---

## Chapter 14. Using the Fault Analyzer IDIS subsystem

Fault Analyzer uses a subsystem for services that can otherwise not be performed or which might cause incomplete analysis if not started. This subsystem is known as the IDIS subsystem.

Currently, Fault Analyzer uses the IDIS subsystem to:

- Connect to DB2 subsystems to read the catalog if the connection failed from the abending address space. These connection failures cause the message  
`IDI0082E DB2 Call Level Interface error: SQL return code -1 for SQLAllocConnect to DB2 system system-id`
- to be issued in the abending address space.
- Perform SVC dump registration when the IDIXTSEL post-dump exit is installed. This registration is primarily intended for CICS system dumps and to facilitate the capturing of Java faults.
- Perform history file access management if PDSESHARING(NORMAL) is used, in order to reduce abend S213-70 cross-system sharing conflicts.
- Optionally, manage history file \$\$INDEX members for improved performance. While this feature is the default for eligible history files, it can be disabled by specifying the NOUPDINDEX PARM field option for the IDIS subsystem, as described in “Starting the IDIS subsystem” on page 241. For details, see “Caching of history file \$\$INDEX data” on page 240.
- Optionally, enable IMS fast duplicate fault suppression specified using the NoDup(ImageFast(*minutes,IMS(...)*)) option. While this feature is the default, it can be disabled by specifying the NOUPDINDEX or NOIMAGEFAST PARM field option for the IDIS subsystem, as described in “Starting the IDIS subsystem” on page 241. For details of the NoDup(ImageFast(*minutes,IMS(...)*)) option, see “NoDup” on page 482.
- Optionally, enable fast Exclude options processing. While this feature is the default, it can be disabled by specifying the NOFASTEXCLUDE PARM field option for the IDIS subsystem, as described in “Starting the IDIS subsystem” on page 241. For details, see “Fast Exclude options processing” on page 282.
- Provide recovery fault recording support. (For general information about recovery fault recording, see “Recovery fault recording” on page 30.)
- Provide improved performance of message and abend code explanation retrieval for the Fault Analyzer ISPF interface LOOKUP command by caching information in storage. Additionally, perform automatic updates of the VSAM KSDS message and abend code explanation repository when required—refer to “Setting up the message and abend code explanation repository” on page 233 for information about required access authorization.
- Provide Java analysis support by means of the Java-supplied Diagnostic Tooling Framework for Java (DTFJ). The DTFJ process runs from the BXPAS address space, which is spawned from the IDIS subsystem when the PARM='JAVA' option is used in the IDIS subsystem startup JCL.
- Perform reading of message and abend explanations from source data sets, as well as caching of selected explanations for improved performance.

The Fault Analyzer IDIS subsystem should not be prioritized lower than any of the tasks for which it might be invoked. Assigning a high priority to the IDIS

## Using the Fault Analyzer IDIS subsystem

subsystem does not affect system performance adversely, since no resources are consumed by the IDIS subsystem when it is not in use.

A separate IDIS subsystem is required on each MVS image running Fault Analyzer.

---

## Sysplex-wide subsystem inter-communication

Multiple IDIS subsystems in a sysplex interface with one another using the Cross-system Coupling Facility (XCF) to allow efficient sharing of subsystem-managed data, such as the caching of \$\$INDEX data. No special action is necessary in order to use this feature.

If the XCF should become unavailable, then IDIS subsystem processing continues to perform all functions, although possibly with reduced performance as a result.

---

## Caching of history file \$\$INDEX data

The default value for the PARM parameter in the IDIS subsystem startup JCL is "UPDINDEX" (see "Starting the IDIS subsystem" on page 241). When the default value is used, the IDIS subsystem manages the \$\$INDEX member access of all PDSE history files:

- That are used on the MVS image where the IDIS subsystem is running.
- To which the IDIS subsystem has UPDATE access.

For details about the \$\$INDEX member, see "Special members in the history file data set" on page 9. PDS-format history files are not managed in the Fault Analyzer IDIS subsystem because their enqueue and parallel update limitations prevent any effective caching of their \$\$INDEX members.

After the \$\$INDEX member from a history file is read, the information is kept in the IDIS subsystem address space during periods of high activity. This approach provides fast access for any requesters of this data, since no I/O is required. Real-time analysis, interactive and batch reanalysis, and the Fault Analyzer ISPF interface, use the cached IDIS subsystem data.

Each time the cache is accessed for read or write, the time limit for in-storage retention is reset. The reset ensures that a history file that is highly active continues to provide fast cache access. This approach is beneficial to environments, such as CICS, where multiple abends might occur in rapid succession, and often all need to update the same history file.

The time limit for the IDIS subsystem in-storage retention is set to 5 minutes. The \$\$INDEX member is written back to the history file and the IDIS subsystem relinquishes control of it when:

- The time limit expires.
- A request for update of the same history file is pending from another MVS image in the same sysplex.

Make sure the IDIS subsystem PARM='UPDINDEX' or PARM='NOUPDINDEX' option is the same for all IDIS subsystems in the sysplex. For details about sysplex sharing of history files, see "Sharing of history files across a sysplex" on page 272.

The UPDINDEX option can be used to ensure that certain low-priority jobs do not cause excessive delays in the Fault Analyzer execution due to serialization of the

history file \$\$INDEX member. These jobs are the jobs that share history files with performance sensitive jobs or execution environments, such as IMS and CICS, in a CPU constrained environment.

If you use the UPDINDEX option, ensure that the Fault Analyzer IDIS subsystem has UPDATE access to all history files that it is expected to handle. No further attempt to manage the \$\$INDEX member of a history file is made until the IDIS subsystem is stopped and restarted:

- After the IDIS subsystem is called for a history file to which it does not have UPDATE access.
- After the IDIS subsystem fails to update a history file for any reason.

**Note:** For maximum Fault Analyzer performance, consider the DeferredReport option. For details, see “DeferredReport” on page 463.

---

## Starting the IDIS subsystem

To start the IDIS subsystem, a simple job as shown below can be submitted:

```
//IDISS JOB ...
//IDISST EXEC PGM=IDISAMAN,TIME=NOLIMIT,REGION=region-size,PARM='options'
//IDIDOC2 DD DISP=SHR,DSN=IDI.SIDIDOC2
//* (Optional DD statements might follow, as described below)
```

where

### **REGION=***region-size*

Specifies the region size to be used for the IDIS subsystem.

In most cases, a region size of 100 megabytes should be adequate (REGION=100M). However, if a very large number of history files are being managed by the IDIS subsystem, or if the history file sizes are very large, then it might be necessary to specify an even larger region size. For information about how to estimate the required region size, see “IDIS subsystem storage requirements” on page 243.

### **PARM=***'options'*

Specifies special options that are only used by the IDIS subsystem to disable some subsystem functions. Further options processing in the subsystem occurs through the standard IDICNFxx parmlib member and the IDIOPTS DD statement, as described in Chapter 33, “Options,” on page 451. The optional PARM field specification can contain one of the following values for *options*:

#### **UPDINDEX**

Specifies that the IDIS subsystem is to manage the \$\$INDEX member access of all PDSE history files used on the same MVS image as where the IDIS subsystem is running, and to which the IDIS subsystem has UPDATE access. For details, see “Caching of history file \$\$INDEX data” on page 240.

This value is the default.

#### **NOUPDINDEX**

Specifies that the abending job performs all history file updates.

#### **IMAGEFAST**

Enables IMS fast duplicate fault suppression specified using the NoDup(ImageFast(*minutes*,IMS(...))) option. For details of the NoDup(ImageFast(*minutes*,IMS(...))) option, see “NoDup” on page 482.

## Starting the IDIS subsystem

This value is the default.

### **NOIMAGEFAST**

Disables IMS fast duplicate fault suppression, regardless of NoDup(ImageFast(...)) settings.

### **FASTEXCLUDE**

Enables fast Exclude options processing. For details, see “Fast Exclude options processing” on page 282.

This value is the default.

### **NOFASTEXCLUDE**

Disables fast Exclude options processing to revert back to normal Exclude processing by the IDIDA task.

### **XCFGGRPSUFFIX=c**

Provides control over the last character, *c*, of the IDIS subsystem XCF messaging group name, IDISXCFc, in order to create an alternative XCF message group. This creation would normally only be done if there are MVS images in a sysplex which do not share DASD and history files with the main IDISXCFM default group, and would be set for the image(s) not sharing DASD and history files. Each messaging group shares updates to the history files by data set name using the XCF messaging group.

### **NOXCFGGRPSUFFIX**

Use the default "M" suffix.

This value is the default.

**JAVA** Enables Java analysis. See “IDIS subsystem requirements for Java” on page 244 for more information.

### **NOJAVA**

Disables Java analysis.

This value is the default.

Multiple PARM field options must be delimited by one or more blank characters.

Alternatively, the IDIS subsystem can be established using a started task. The IDIS subsystem dynamically allocates data sets to SYSOUT=\*, so it must be run under the job entry subsystem (JES).

**Note:** Ensure that the TIME=NOLIMIT parameter is specified as shown in the example above to prevent IDIS subsystem abend S522.

The subsystem name that is used by Fault Analyzer for this subsystem is IDIS. This name does not need to be defined in the IEFSSNxx parmlib member as it is dynamically defined by the IDISAMAN program.

The IDIDOC2 DD statement is used to allocate the IDI.SIDIDOC2 SMP/E target data set, which might contain updates to the Fault Analyzer VSAM KSDS message and abend code explanation repository. If updates are available, and either the IDIDOC2 DD statement has not been specified, or the high-level qualifier of the IDI.SIDIDOC2 data set is not the same as the high-level qualifier of the IDI.SIDIDOC1 data set identified via the DataSets(IDIDOC(...)) option, then message IDI0165A is issued. If this situation occurs, then add the IDIDOC2 DD statement as shown above.

## IDIS subsystem storage requirements

If using PARM='NOUPDINDEX', then there is no requirement for any REGION size specification—a default 32 MB region is adequate.

If using PARM='UPDINDEX' (this value is the default), then a larger region size should be used. The region size in megabytes can be approximated as follows:

$$32 + (\text{num\_hist\_files} * (0.5 + (\text{avg\_num\_entries} * 0.0005)))$$

where:

### **num\_hist\_files**

The total number of PDSE history files that are managed by the IDIS subsystem.

### **avg\_num\_entries**

The average number of fault entries in the managed history files.

If a maximum number of fault entries has been set for a history file using the IDIUTIL batch utility SetMaxFaultEntries control statement (for details, see Chapter 27, “Managing history files (IDIUTIL utility),” on page 337), then this number is the number of fault entries that should be included for that history file.

## IDIS subsystem requirements for DB2

You must add DD statements for all DB2 subsystems that are not accessible via LINKLIST, and for which you want Fault Analyzer to perform analysis:

- If you have more than one version of DB2 installed.
- If the DB2 load module library is not in LINKLIST.

Here is the DD statement format:

```
//DB2subsystem-id DD DISP=SHR,DSN=data-set-name
```

where *subsystem-id* is the DB2 subsystem ID (usually 4 characters), and *data-set-name* is the associated load module library. For a data sharing group, the group attach name is used as the subsystem ID, regardless of whether the DB2 subsystem is running in data sharing mode or not.

If, for example, the DB2 subsystem with an ID of DSN1 requires the load library DSN1.SDSNLOAD, which is not in LINKLIST, then add the JCL DD statement

```
//DB2DSN1 DD DISP=SHR,DSN=DSN1.SDSNLOAD
```

to the Fault Analyzer IDIS subsystem job.

Even if only a single version of DB2 is installed and available from LINKLIST, then you might still want to add a DD statement for the DB2 load library since message IEC130I might otherwise be issued due to missing DDname. Note that Fault Analyzer processing is unaffected by this message.

Do not include a DSNAOINI DD statement in the IDIS subsystem JCL, as this DDname is allocated dynamically by Fault Analyzer as needed for the appropriate DB2 subsystem.

The IDIS subsystem needs EXECUTE access to the DSNACLI plan, and SELECT authority to the following SYSIBM catalog tables:

SYSIBM.SYSDBRM  
SYSIBM.SYSPACKAGE

## Starting the IDIS subsystem

```
SYSIBM.SYSPACKSTMT  
SYSIBM.SYSPACKLIST  
SYSIBM.SYSPLAN  
SYSIBM.SYSSTMT
```

## IDIS subsystem requirements for Java

Ensure that data set IDI.SIDIAUT2 has been added to LINKLIST (for details, see “Making Fault Analyzer modules available” on page 231). Failure to add IDI.SIDIAUT2 to LINKLIST prevents Fault Analyzer from loading DLLs which are necessary for the analysis of Java faults. Also, message IDI0158W is issued by the IDIS subsystem if this situation should occur.

The IDIS subsystem must have UPDATE access to at least the default history file in order to create a Java fault entry.

### Specifying an IDIJLIB DD statement

An optional IDIJLIB DD statement can be included in the IDIS subsystem JCL for Java as follows:

```
//IDIJLIB DD PATH='path'
```

The IDIJLIB JCL statement specifies a target directory for HFS executables. The IDIS subsystem writes a small number of program files to this directory as part of its execution, thus the IDIS subsystem must have authority to read, write, and execute files in the specified directory. Additionally, diagnostic information might also be written to this path. The path name is case sensitive and the path must exist. An example of a possible specification is PATH='/u/*user-id*/idij', where *user-id* is the user ID under which the IDIS subsystem is running.

If IDIJLIB is not provided, then the default path for the IDIS subsystem user ID is used.

For each image in a sysplex, Fault Analyzer creates a subdirectory of the IDIJLIB path, using the &SYSCLOSE name. This naming ensures that each image writes to a separate directory.

### Specifying an IDIJVM DD statement

An optional IDIJVM DD statement can be included in the IDIS subsystem JCL for Java as follows:

```
//IDIJVM DD PATH='path'
```

**Note:** Performing Java analysis requires a JVM that has a service level equal to, or greater than, the JVM that recorded the Java dump. This IDIJVM DD statement specifies a default JVM to use for performing Java analysis, when the version of Java used to take the dump cannot be found on the system (based on the JAVA\_HOME path recorded in the dump).

---

## Stopping the IDIS subsystem

The IDIS subsystem can be stopped and restarted at any time. When the IDIS subsystem is inactive (stopped), all Fault Analyzer processes continue to run, except for some DB2 catalog data access. There is also an increase in CPU and I/O usage, compared to having the IDIS subsystem running with PARM='UPDINDEX'.

## Stopping the IDIS subsystem

To permit normal termination, a MODIFY command should be used to stop the IDIS subsystem by, for example, issuing

`F name,STOP`

where *name* is the appropriate identifier for the MODIFY command, depending on the way in which the IDIS subsystem was started.

Alternatively, use

`P name`

Never use the CANCEL command to stop the IDIS subsystem.

If either the Fault Analyzer IDIS subsystem is not active, or if an incorrect identifier was used on the MODIFY command, MVS issues the message:

`IEE341I XYZ NOT ACTIVE`

If the Fault Analyzer IDIS subsystem is already active when another attempt to start it is performed, then the following message are issued to the operator console:

`IDISAMAN The Fault Analyzer Subsystem is already active in jobname job-id`

where *jobname* is the job or started task name of the currently executing Fault Analyzer IDIS subsystem and *job-id* is the JES job or started task ID.



---

## Chapter 15. Modifying your ISPF environment

To use Fault Analyzer with ISPF, you need to ensure that the appropriate data sets have been allocated and that one or more ways to invoke Fault Analyzer have been provided. This process is explained in the following.

---

### Allocating ISPF data sets

The following data sets must be allocated to the respective ISPF DDnames (either in the TSO logon procedure or using any other installation-specific method):

DDname	Data set name
ISPPLIB	IDI.SIDIPLIB
ISPMLIB	IDI.SIDIMLIB
ISPSLIB	IDI.SIDISLIB
ISPTLIB	IDI.SIDITLIB
SYSEXEC	IDI.SIDIEXEC

A sample REXX EXEC that can be used to invoke Fault Analyzer from within ISPF is provided as member IDISISPF in data set IDI.SIDISAM1. The EXEC performs the necessary dynamic definition of the required data sets using the ISPF LIBDEF and TSO ALTLIB services.

**Note:** To enable the editing or browsing of data sets using IBM File Manager for z/OS, all necessary ISPF libraries for File Manager must be made available also when Fault Analyzer is invoked. Refer to the File Manager documentation for information about the required data set names that should be added to your TSO logon procedure or invocation exec. The IDISISPF sample exec allows you to optionally include File Manager data sets.

To help with diagnosis of problems relating to the allocation of data sets for Fault Analyzer, the TSO/ISPF commands ISRDDN, ISRFIND, or ISPLIBD might be useful.

---

### Making the Fault Analyzer IDISCMDS command table available

Fault Analyzer provides a sample command table as member IDISCMDS in data set IDI.SIDITLIB. This command table provides definitions that are required in order to issue the SFA command (see “Invoking Fault Analyzer from SDSF (SFA command)” on page 249), the LOOKC command (see “Invoking the LOOKUP command using cursor selection (LOOKC command)” on page 250), and the LOOKUP command (see “LOOKUP” on page 69).

The IDISCMDS command table must be made available by defining it to ISPF. For information about how to do this, refer to *z/OS ISPF Planning and Customizing*, chapter “Customizing DM”, section “Customizing Command Tables”.

### Updating the ISPF selection panel

Update your ISPF selection panel to include an option for selecting the IDIPDDIR program to start the ISPF history file interface using the IDI application ID. For example, to invoke Fault Analyzer using option 9, update the )PROC section of the ISPF selection panel as follows:

```
)PROC  
&ZSEL = TRANS (TRUNC (&ZCMD,'.')  
.  
. (other selections)  
.9,'PGM(IDIPDDIR) NEWAPPL(IDI) SCRNAME(FAULTA)'  
X,EXIT  
.,.  
*,?' )  
&ZTRAIL=.TRAIL
```

**Note:** The above IDI application ID is shown as an example only. If a different value has been used for an earlier version of Fault Analyzer, then changing it should be avoided since it results in the loss of all ISPF interface user-tailoring.

As an alternative to calling program IDIPDDIR directly from the selection panel as shown above, you might instead consider invoking the IDISISPF sample REXX EXEC. This invocation would eliminate the need to have Fault Analyzer ISPF and TSO libraries allocated prior to entering ISPF.

Remember to also update the )BODY section as appropriate.

As an alternative to the direct invocation of the Fault Analyzer ISPF interface via the ISPF selection panel, a customized front-end can be used. A sample front-end is shown in Appendix B, "Sample customized ISPF interface front-end," on page 569, which can be modified to suit any specific requirements your installation might have.

### Invoking Fault Analyzer using an ISPF 3.4 line command (FA)

There is an alternative to the typical invocation of the Fault Analyzer ISPF interface, which uses an option from an ISPF selection menu. Sometimes it is more convenient to invoke the Fault Analyzer ISPF interface directly as a line command against an MVS dump data set, a history file, or a CICS auxiliary trace data set on the ISPF option 3.4 Data Set List Utility panel. This invocation has the advantage of being able to perform interactive dump analysis, or display fault entries, or CICS trace information using the selected data set immediately. You do not have to initiate the appropriate action by using the File pull-down menu for a dump data set, or by typing the data set name for a history file.

To facilitate this, a REXX exec (named FA, for example) is required in one of the data sets that is allocated to the SYSEXEC concatenation of the ISPF user. This exec name can then be entered next to a dump data set, history file data set name, or a CICS auxiliary trace data set name in an ISPF option 3.4 data set list. The exec invokes Fault Analyzer using the correct ISPF NEWAPPL ID, and passes the name of the data set as a parameter. The exec determines the data set type from its allocation attributes.

The FA exec:

## Invoking Fault Analyzer using an ISPF 3.4 line command (FA)

```
Parse Arg dsn .

If dsn = '' Then
  Do
    Say 'Data set name required.'
    Exit(4)
  End
  /* Determine if dsn is a history */
  /* file or a MVS DUMP           */

outl. =
x = Outtrap('OUTL.',,"NOCONCAT")
Address TSO "LISTDS " || dsn
x = Outtrap(OFF)

svcdump = 0
auxtrac = 0
If outl.0 > 2 Then
  Do
    lrecl = Word(outl.3,2)
    If lrecl = 4160 Then
      svcdump = 1
    Else
      If lrecl = 4096 Then
        auxtrac = 1
  End
  /* Invoke the Fault Analyzer      */
  /* ISPF interface passing the    */
  /* appropriate PARM string.      */

Address ISPEXEC
If svcdump = 1 Then
  'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) PARM(DSN('dsn'))'
Else
  If auxtrac = 1 Then
    'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) PARM(AUXTRACEDSN('dsn'))'
  Else
    'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) PARM(ISPFHISTDSN('dsn'))'
Exit
```

This example is included in data set IDI.SIDIEEXEC as member IDISFA.

While IDISFA works as a line command against ISPF 3.4 data sets, as long as IDI.SIDIEEXEC is included in the SYSEXEC concatenation, it might be more convenient to copy this exec to another data set in the SYSEXEC concatenation with a shorter name, for example FA.

---

## Invoking Fault Analyzer from SDSF (SFA command)

An exec named IDISSDSF is provided in data set IDI.SIDIEEXEC. This exec can be used to extract the history file data set name and the fault ID from message IDI0003I, while browsing a job or the syslog in SDSF under ISPF, and then invoke Fault Analyzer for interactive reanalysis of that fault.

The IDISSDSF exec is mapped to an ISPF command called SFA in the Fault Analyzer IDIS ISPF command table IDISCMDS, which must be made available to ISPF—for details, see “[Making the Fault Analyzer IDISCMDS command table available](#)” on page 247.

## **Invoking the LOOKUP command using cursor selection (LOOKC command)**

An exec named IDISMLKP is provided in data set IDI.SIDIEEXEC. This exec can be used to determine the word at the current cursor position, and then to invoke the Fault Analyzer LOOKUP command, passing the cursor word.

The IDISMLKP exec is mapped to an ISPF command called LOOKC in the Fault Analyzer IDIS ISPF command table IDISCMDS, which must be made available to ISPF—for details, see “Making the Fault Analyzer IDISCMDS command table available” on page 247.

---

## **Providing ISPF interface defaults for new users**

To provide installation-specific defaults for new users of the Fault Analyzer ISPF interface, do the following:

1. Invoke the Fault Analyzer ISPF interface.
2. Set options to the values that new users should see when first using this interface. These values might include:
  - The initial history file or view selected from the Fault Entry List display.
  - Fault Analyzer preferences (from the "Options" pull-down menu).
  - Batch reanalysis options (from the "Options" pull-down menu).
  - Interactive reanalysis options (from the "Options" pull-down menu).
  - View settings (from the "View" pull-down menu).
3. Exit from the Fault Analyzer ISPF interface.
4. Copy the *applidPROF* member, where *applid* is the application identifier used for Fault Analyzer in your installation (for example, IDI), from your ISPF profile data set to a data set that must be made available to all users in their ISPTLIB concatenation.

**Note:** It is recommended that any installation-wide defaults for the columns shown on the Fault Entry List display are provided by the specification of the HistCols option in the IDICNF00 parmlib member, rather than defining them through the ISPF profile member defaults. This approach is necessary because users, who might make other changes to these columns, are otherwise not able to reset the columns to the installation-wide defaults by pressing PF4.

As soon as a user has made changes to any of the variables contained in the profile member, it is saved in their private profile data set identified by the ISPPROF DDname, and read from there on any subsequent use of the Fault Analyzer ISPF interface.

---

## **Providing installation-specific batch reanalysis JCL control statements**

Whenever a user performs batch reanalysis of a history file entry, using the B line command from the Fault Entry List display, Fault Analyzer generates the appropriate JCL stream based on options that are specified on the user's Batch Reanalysis Options display. To permit installations to always add their own JCL control statements to such generated jobs, Fault Analyzer provides support for an optional user-provided skeleton member. If found, this member is inserted immediately following the JOB card of the generated JCL stream. The skeleton

## **Providing installation-specific batch reanalysis JCL control statements**

member must be named IDISJCTL, and be available from the ISPSLIB concatenation. If it does not exist, or is not found, then it is simply not used.

A possible use of this member is to add a JCL control card to permit more than the default number of output lines. For example:

```
/* Override of installation default output line limit  
/*JOBPARM LINES=300
```

---

## **Increasing the display area for Fault Analyzer reports**

ISPF supports screen sizes with a width greater than 80. By allowing Fault Analyzer to use a wider screen, readability of online reports is greatly improved and the necessity of using left and right scrolling commands can be eliminated.

Here is the process:

1. Check that you have met the technical requirements that are set out in Appendix D, "Technical details for screen size adjustments," on page 581.
2. Alter the emulator settings to use, for example 27x132.
3. Logon to ISPF.
4. Go to ISPF Option 0. Entering SETTINGS on any command line displays the same screen.
5. Set the screen format to DATA or MAX. See the relevant ISPF publications for explanations of the difference in behavior of DATA and MAX.

## **Increasing the display area for Fault Analyzer reports**

---

## Chapter 16. Customize Fault Analyzer by using USERMODs

Some aspects of the Fault Analyzer customization can only be performed via SMP/E USERMODs, as described in the following, while others can be performed using an IDIOPTLM configuration-options module (for details, see Chapter 17, “Customize Fault Analyzer by using an IDIOPTLM configuration-options module,” on page 257).

---

### Enabling Fault Analyzer to be invoked

In order for Fault Analyzer to be given control when a program abends, a number of invocation exits must be installed.

The following provides information about the installation of the two invocation exits that are required for non-CICS environments. Refer to Chapter 21, “Customizing the CICS environment,” on page 301 for invocation exits for the CICS environment.

### Installing the MVS change options/suppress dump exit IDIXDCAP

The installation of the Fault Analyzer dump capture exit, IDIXDCAP, depends on the level of z/OS:

#### For z/OS 2.2 and later levels of z/OS

IDIXDCAP is installed as a IEAVTABX\_EXIT dynamic exit routine by including the following in a PARMLIB PROGxx member, which is automatically selected at IPL time:

```
EXIT ADD EXITNAME(IEAVTABX_EXIT) MODNAME(IDIXDCAP)
```

To activate the IDIXDCAP exit prior to the next IPL, issue the operator command:

```
SET PROG=xx
```

where xx matches the suffix used on the PARMLIB PROGxx member.

#### For levels of z/OS prior to z/OS 2.2

Fault Analyzer requires a dump capture exit, IDIXDCAP, to be installed in the IEAVTABX installation exit list. This exit is installed by the IDITABX USERMOD.

To install this USERMOD, edit and submit the sample job IDITABX. This job includes IDIXDCAP in the IEAVTABX installation exit list. If you have other exits defined in this list, add the IDIXDCAP exit last.

For more information about adding exits to IEAVTABX, see *MVS Installation Exits*.

To activate this change, do one of the following:

- Issue the command:

```
SETPROG LPA,ADD,MOD=(IEAVTABX),DSN=SYS1.LPALIB
```

**Note:** Refer to the output from the SMP/E APPLY for information about the data set containing the updated IEAVTABX load module, in case it is not the usual SYS1.LPALIB.

## Enabling Fault Analyzer to be invoked

- Re-IPL with CLPA.

## Enabling the Language Environment abnormal termination exit IDIXCEE

Fault Analyzer requires a Language Environment abnormal termination exit, IDIXCEE, as an extra method of invoking Fault Analyzer to the MVS change options/suppress dump exit. To enable this exit, you must add it to the CEEEXTAN CSECT for Language Environment. To do this, make a copy of the CEEWDEXT softcopy sample member in the CEE.SCEESAMP data set. Make the changes suggested in the sample member and replace the lines:

```
<<< REPLACE THESE 2 LINES WITH A COPY OF CEEEXTAN  
AND OVERRIDE AS DESIRED >>>
```

with

```
CEEEXAHD      ,User exit header  
CEEART TERMXIT=IDIXCEE  
CEEAST       ,Terminate the list
```

If more than one exit is specified in the CEEEXTAN list (for example, due to other similar third-party products also being used), then Fault Analyzer should be the first exit specified.

If an IPL of MVS is not planned as part of the Fault Analyzer installation, then this exit can be activated as follows, depending on whether LE has been placed in LPA or not:

- **LE in LPA**

Determine the data set name and load modules updated by the USERMOD from the APPLY job output and issue the command:

```
SETPROG LPA,ADD,MOD=(module-list),DSN=data-set-name
```

If one or more of the updated load modules are not in LPA, then continue with the instructions for "LE not in LPA" below.

**Note:** Remember to reissue the SETPROG command after the next IPL, unless the IPL is performed with the CLPA option.

- **LE not in LPA**

If LE is not in LPA, it is assumed to be in LINKLIST. To activate the exit, issue the command:

```
F LLA,REFRESH
```

If you prefer to selectively refresh only the affected load modules, then you need to check the SMP/E APPLY output for the updated data sets and load module names, add these to a CSVLLAxx parmlib member using the LIBRARIES and MEMBERS parameters, and then issue the command F LLA,UPDATE=xx.

Alternatively, use Data Set Commander for z/OS to perform the refresh.

For general information about implementing an Language Environment abnormal termination exit, refer to the *Language Environment Customization* book.

**Note:** The CICS version of this exit, CEECXTAN, which is installed with sample job CEEWCEXT in data set CEE.SCEESAMP, is very similar to this exit. If you are going to install both, make sure that you double-check that you have used the correct names and not installed the same exit twice.

---

## **Working with applications that use a non-Language Environment run time**

The following USERMODs are applicable to non-LE applications only.

### **Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++IDISPLI/++IDISPLIA)**

To have Fault Analyzer invoked implicitly for abends occurring in applications using the PL/I version 2 release 3 non-LE runtime library, the IDISPLI or IDISPLIA USERMOD must be applied. Either USERMOD modifies the ONCODE processing in the PL/I runtime load module IBMBLIIA to call the Fault Analyzer program SNAP interface (IDISNAP). However, while the IDISPLI USERMOD returns to PL/I after the call to IDISNAP, the IDISPLIA USERMOD issues a deliberate abend U3001, which can make it more suitable for environments, such as IMS, where ROLLBACK might be required.

This USERMOD is not required if using the LE run time.

To apply this USERMOD, edit and submit the sample job IDISPLI or IDISPLIA.

### **Always invoking Fault Analyzer from PL/I PLIDUMP (++IDISPDM)**

If PL/I applications have been written to handle errors by calling PLIDUMP with the 'S' option, then Fault Analyzer cannot be invoked (see "Application-handled error conditions" on page 229). For this reason, a USERMOD is provided that modifies the PL/I PLIDUMP main control routine to always invoke Fault Analyzer using IDISNAP, ahead of normal PLIDUMP processing.

To apply this USERMOD, edit and submit the sample job IDISPDM.

---

### **Eliminating the need for a dump DD statement (++IDITABD)**

| **Note:** This ++USERMOD is not applicable to z/OS 2.2 or later.

The IDITABD USERMOD is required for all non-LE batch jobs, for which fault analysis is to occur, unless either a SYSABEND, SYSUDUMP, or SYSDMDUMP DD statement is used. This requirement is also true for all LE batch jobs with TERMTHDACT(UA\*) in effect, if the IDIXCEE Language Environment exit is not installed.

The USERMOD permits the IEAVTABX-defined exits to be invoked (which includes the Fault Analyzer change options/suppress dump exit, IDIXDCAP), regardless of any JCL dump DD statement specification.

To apply the IDITABD USERMOD, edit and submit the sample job IDITABD.

To activate this change, IPL again with CLPA. (It is not possible to activate this change using the SETPROG command.)

### Obtaining load modules from CA-Panexec

In order to obtain CSECT information for load modules that are managed by CA-Panexec, Fault Analyzer provides the ability for users to install an exit that can make this possible. Normally, when load modules are managed by CA-Panexec, customers will see IEW2717S and IEW2718S I/O errors as the IBM Binder program attempts to include the load module from the CA-Panexec-managed library. As a result, Fault Analyzer might not be able to provide source line information for the point of failure.

The function of the CA-Panexec exit is to copy the load module from the CA-Panexec-managed data set to a temporary data set in normal load module format which the Binder can use.

To install an exit for this purpose, the IDILMODX CSECT in module IDIDA, which consists of 8 bytes that are normally blank, can be zapped by the customer to provide the name of a load module that is to be called prior to Binder INCLUDE processing. The load module named in IDILMODX should be accessible via an MVS LOAD macro.

The IDILMODX-named exit is passed a parameter list in register 1, addressing two pointers to two 8 byte fields. The first field contains the name of the load module to be examined (the load module the Binder includes to extract CSECT data), while the second 8 byte field is a DD name, or blank. If the DD name is not blank, then it indicates that this DD is intended to be used for the Binder in searching for the load module name. Fault Analyzer normally only sets the DD name field to DFHRPL when running under CICS, on most other occasions the DD name is blank, indicating Fault Analyzer leaves it to the Binder to locate the data set from which the module should be read. The blank DD contains X'00'. The IDILMODX-named exit is permitted to update the second parameter to a new DD name from which the Binder should read the module, or to leave the DD name unchanged. If the DD name is updated, then it is used for the Binder INCLUDE call of the module.

A sample CA-Panexec exit is available as member IDIPANEX in data set IDI.SIDISAM1.

### Using a non-standard LE parameter list separator character (++IDIOPT1)

If you have changed the standard forward slash (“/”) Language Environment parameter list separator character, then you need to install the Fault Analyzer IDIOPT1 USERMOD.

To apply this USERMOD, edit and submit sample job IDISOPT1.

### Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (++IDIOPT2)

To suppress Fault Analyzer real-time analysis via the IDIXDCAP pre-dump invocation exit, if a prior pre-dump exit has set RC=8 to suppress the dump, install the IDIOPT2 USERMOD.

To apply this USERMOD, edit and submit sample job IDISOPT2.

---

## Chapter 17. Customize Fault Analyzer by using an IDIOPTLM configuration-options module

Some aspects of the Fault Analyzer customization can only be performed via SMP/E USERMODs (for details, see Chapter 16, “Customize Fault Analyzer by using USERMODs,” on page 253), while others must be performed by using an IDIOPTLM configuration-options module.

An IDIOPTLM configuration-options module can be used to provide the following configuration settings:

### LEDSN

For details, see “Identifying the LE runtime library.”

### CNFDSN

For details, see “Specifying an alternative parmlib data set for IDICNFxx” on page 258.

### RFRDSN

For details, see “Changing the default recovery fault recording IEATDUMP data set name” on page 258.

### CICSNOA

For details, see “Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit” on page 258.

A sample job to create an IDIOPTLM configuration-options load module is provided as member IDIOPTLM in data set IDI.SIDISAM1.

To specify either of the above settings, edit and submit the IDIOPTLM sample job.

The name of the configuration-options load module must be IDIOPTLM, and it must be placed in an APF-authorized library in order to be used. The library should also be in LNKLST so that the IDIOPTLM load module can be found by all jobs. IBM recommends the use of IDI.SIDIAUTH for this authorized library.

---

## Identifying the LE runtime library

The LEDSN setting in the IDIOPTLM configuration-options module must identify the LE runtime library data set name (which must be APF authorized) when all of these conditions apply:

- You run Fault Analyzer against abends that occur in a non-LE environment.
- LE is not in LINKLIST.
- The LE runtime library data set name is not CEE.SCEERUN.

If LE is in LINKLIST, or if the LE runtime data set name is CEE.SCEERUN, then the LEDSN setting need not be used.

The LEDSN setting applies to Fault Analyzer real-time execution that is using the CICS XPCABND global user exit (IDIXCX52 or IDIXCX53) or the IEAVTABX MVS change options/suppress dump exit (IDIXDCAP) only. It does not apply to the Fault Analyzer ISPF interface.

---

## **Specifying an alternative parmlib data set for IDICNFxx**

To accommodate installations that do not provide general READ access to SYS1.PARMLIB (or any one of the data sets in the logical parmlib concatenation), Fault Analyzer provides the CNFDSN setting in the IDIOPTLM configuration-options module. This setting permits an alternative data set to be specified for the IDICNFxx configuration member.

The specified data set name can contain MVS system symbols.

Because Fault Analyzer also needs to read the IBM Problem Determination Tools for z/OS Common Component IPVCNF00 configuration member, an installation requiring the use of IDIOPTLM to specify an alternative data set for IDICNFxx, similarly requires the use of IPVOPTLM to specify an alternative data set for IPVCNF00. For information about the IPVOPTLM configuration-options module and the IPVCNF00 parmlib member, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

---

## **Changing the default recovery fault recording IEATDUMP data set name**

In situations where an SDUMP cannot be used, then the Fault Analyzer recovery fault recording feature requires that an IEATDUMP data set can be allocated by the abnormally terminating real-time analysis task. (For general information about the recovery fault recording feature, see “Recovery fault recording” on page 30.)

The data set name to be used is by default set to:

`IDIRFRHQ.IDIRFR.&SYSNAME..D&YYMMDD..T&HHMMSS..&JOBNAME.`

To change the default data set name, the RFRDSN setting in the IDIOPTLM configuration-options module can be used.

Like the default data set name provided, the data set name that is specified using the RFRDSN setting should contain MVS system symbols to ensure that each allocated data set is unique.

**Note:** It is important to ensure that all users are able to allocate data sets with the name specified—see “Managing recovery fault recording data set access” on page 234 for more information.

If it is desirable that the recovery fault recording IEATDUMP data set name contains local date and time instead of UTC, then change &YYMMDD to &LYYMMDD and &HHMMSS to &LHHMMSS.

---

## **Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit**

The CICSNOA setting in the IDIOPTLM configuration-options module can be used to change the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit, and should only be considered if an installation requires special handling of CICS transaction abends as outlined in the following.

## **Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit**

When a prior CICS exit program has specified a return code UERCBYP (suppress transaction dump), then Fault Analyzer by default continues to perform analysis. This behavior is equivalent to the way in which non-CICS abends are handled by the MVS pre-dump exit, IDIXDCAP.

When using the CICSNOA setting, a prior XDUREQ exit program return code of UERCBYP stops Fault Analyzer from performing analysis.



---

## Chapter 18. Setting up history files

Fault Analyzer requires at least one PDS or PDSE data set to be allocated as a fault history file.<sup>8</sup>

---

### Determining what size history files to allocate

The size of a history file to be allocated depends on several factors, such as the installation environment and the types of jobs for which faults are being recorded. However, the following can be used as an initial approximation of the space requirement expressed in kilobytes:

$$\text{kilobytes} = 500 * \text{number-of-entries}$$

where:

*number-of-entries*

The maximum number of concurrent entries that you want to hold in the history file.

A more accurate determination of the average fault entry size can be made by dividing the actual history file space utilization by the total number of members after having recorded a representative number of faults.

---

### Allocating a PDS or PDSE for a history file

It is recommended that PDSE data sets are used, although you can also use PDS-managed (not recommended) history files. This recommendation is because of PDSE data set automatic space management and superior directory integrity for shared usage. If your environment does not yet use PDSE data sets because System Managed Storage is not yet implemented in your MVS image, you might find it worth while installing the MVS APARs mentioned in information APAR II12221. APAR II12221 lists the MVS APARs required to make PDSE support available without implementing System Managed Storage.

If using PDSE version 2 data sets, then do not define history files with member generation support.

To allocate one or more history files, edit and submit the sample job IDISHIST. This job allocates two PDSE data sets named IDI.HIST and IDI.HIST.TEST respectively. All users whose abending jobs are recorded in these history files need write access to the data sets.

Refer to the instructions in the sample job for information about changes you might need to make to this job.

As an alternative to submitting a batch job to allocate a new history file, one can be allocated by selecting the Fault Entry List display action-bar File pull-down menu. For details, see “New history file allocation” on page 55.

---

8. Although DUMMY specification of the history file is supported, it is recommended that an actual data set is used in order to take advantage of the features provided with the ISPF interface.

## Allocating a PDS or PDSE for a history file

It is important that the history file is allocated with sufficient space to accommodate all data that is expected to be written to it. Otherwise, out-of-space conditions might occur that cause Fault Analyzer to abend (for example, system abend SE37).

To assist with the history file space management, Fault Analyzer provides the IDIUTIL batch utility, which (among others) can be used to:

- Delete old history file entries (DELETE control statement).
- To set up a self-maintaining history file (SetMinFaultEntries control statement). See “AUTO-managed PDSE history files” for more information about AUTO-managed PDSE history files.

For details on the IDIUTIL batch utility, refer to Chapter 27, “Managing history files (IDIUTIL utility),” on page 337.

The above IDIUTIL batch utility functions can alternatively be performed using the Fault Analyzer ISPF interface (for details, see “Change fault history file settings” on page 57).

In a sysplex, history files can be shared between any number of systems. For more information about history files in a sysplex, see “Sharing of history files across a sysplex” on page 272.

---

## AUTO-managed PDSE history files

By default, a newly created PDSE history file is AUTO-managed. Alternatively, this management can be controlled using the IDIUTIL batch utility SetMinFaultEntries control statement, as described in “SETMINFAULTENTRIES control statement” on page 342, or by using the Fault Analyzer ISPF interface (for details, see “Change fault history file settings” on page 57).

A history file is not actively AUTO-managed until a minimum number of fault entries have been created. The minimum number of fault entries is by default 25. Change the value by using the IDIUTIL batch utility SetMinFaultEntries control statement, or by using the Fault Analyzer ISPF interface. Until the minimum number of fault entries have been created, the history file data set space is permitted to increase by allocation of secondary extents.

Once more than the minimum number of fault entries exist, then Fault Analyzer deletes the oldest eligible fault entries, in excess of 25, to provide the space required for writing new fault entries. This process generally prevents the allocation of more data set extents and related out-of-space conditions.

For information about the assignment and reuse of fault IDs, see “Fault history files” on page 8.

---

## Providing the name of a history file to Fault Analyzer

The current fault history file is determined by an explicitly coded IDIHIST DD statement in the abending job step or the specification of the IDIHIST suboption of the DataSets option. If neither of these are found, the default name is IDI.HIST.

## Setting up views

This section explains how to set up views for an installation—for general information about using views, see “Using views” on page 37.

A view is essentially a list of history files, that are all displayed together using the Fault Analyzer ISPF interface.

If not using the XFACILIT class to provide access to individual history file fault entries (see “Managing history file fault entry access” on page 273), then there might be special considerations required in order to decide on the kind of views an installation want to create. These are outlined in “View considerations if not using XFACILIT resource class” on page 265.

The list of history file data set names to make up a view are simply placed into a member in another PDS that was created specifically to hold the view definitions. This member is generally an installation-wide data set to which all users have read access. It is pointed to by the IDIVIEWS data sets in the DataSets option. The names of the members are used as the view names. The users are given a list of the view names when using the ISPF options pull down to change the history file and pressing F4 to list views. The first line of these members can be an optional comment starting with an asterisk in column one. If this comment line exists in a view member it is displayed along side of the view name in the selection screen. The comment line can provide a description of the view to assist the users choice.

Figure 129 shows an example of a view data set member.

---

```
* Department P024 history files 1
P024.&SYSNAME..CICS.HIST 2
P024.IMS.HIST 3
* Include the installation-wide dehistory file 4
IDI.HIST 5
```

---

*Figure 129. Sample view data set member*

In the above example:

- 1** This line is a comment (asterisk in column 1). Because this comment is on the first line of the view member, it is also used as the view description when displaying the list of views using the Fault Analyzer ISPF interface.
- 2, 3 and 5** These are the fully qualified history file names that is displayed simultaneously through the Fault Analyzer ISPF interface when selecting this view. All data set names must be specified on a single line each, starting in column 1. Surrounding the data set names in single quotes is optional. There is no limit on the total number of data set names that can be specified in a single view.
- The symbol substitution rules that apply to the data set names that are specified in a view member are the same as those that apply to the data set names that are specified in the IDIVIEWS suboption of the DataSets option. See “Data set name substitution symbols” on page 460 for more information.
- 4** This line is a comment that is ignored by the Fault Analyzer ISPF interface.

## Setting up views

No specific view data set attributes are required, except that it must be a PDS(E). The logical record length must be large enough to contain the longest data set name and the longest -HistCols (see “Specifying a default column layout”) or -Match (see “Specifying an initial fault entry selection criteria”) specification. As an example, use record format VB with a logical record length of 32,756 bytes. The view member must not contain sequence numbers.

For easier reference to fault IDs across multiple history files, the ability to set up to three alphabetic fault prefix characters for individual history files is available. Using different prefix characters between different groups helps users recognize the owning group when viewing faults in a composite view display. The fault prefix characters can be set or changed using the IDIUTIL batch utility SETFAULTPREFIX command.

## Specifying a default column layout

A default column layout can be specified in a view using the -HistCols keyword in column 1 of any record in a view. However, avoid specifying this on the first record if a view description is also desired.

Following the -HistCols keyword, must be a list of Fault Entry List display columns enclosed in parenthesis. The syntax and format of the column name list is the same as for the HistCols option (see “HistCols” on page 479). An easy way to obtain the column name list is to cut and paste from an actual Fault Entry List display where the columns have been configured as desired from the Fault Entry List Column Configuration display (see “Fault entry list column configuration” on page 43).

The complete -HistCols specification must fit on a single view member record—continuation over multiple records is not supported.

No part of the -HistCols specification is case sensitive, including the keyword itself, and any number of blank characters, or none, can precede the open parenthesis following the keyword.

If multiple -HistCols specifications are found in a single view member, then only the last one takes effect.

Figure 130 shows an example of a view data set member specifying a default column layout.

---

```
* All IMS history files
-HistCols(IMS_Pgm Jobname User_ID System Abend Date      Time      )
SYS1.IMS.HIST
SYS2.IMS.HIST
```

---

Figure 130. Sample view with default column layout specification

## Specifying an initial fault entry selection criteria

An initial fault entry selection criteria can be specified in a view using the -Match keyword in column 1 of any record in a view. However, avoid specifying this on the first record if a view description is also desired.

**Syntax**

```

    match (column_name=column_value)
  
```

The *column\_value* permits the use of wildcards. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character. Only table data rows that match the specified string are shown.

The complete -Match specification must fit on a single view member record—continuation over multiple records is not supported.

No part of the -Match specification is case sensitive, including the keyword itself, and any number of blank characters, or none, can precede the open parenthesis following the keyword.

If multiple -Match specifications are found in a single view member, then only the last one takes effect.

Figure 131 shows an example of a view data set member specifying an initial fault entry list criteria.

---

```

* My CICS view
-Match(CICS_Trn *)
PROD.CICS.HIST
  
```

---

*Figure 131. Sample view with initial fault entry list criteria*

## View considerations if not using XFACILIT resource class

To do the most effective setup of views for an installation which is not using the XFACILIT resource class to control history file fault entry access, an understanding of all of the different sets of users of Fault Analyzer is required. The main aspects of each user group that need to be determined are:

1. Their data set security profiles for data sets with write access.
2. The need to review or work on faults from jobs submitted by other user groups.
3. The need for read access only, or no access at all, to fault entries between groups.

These requirements are already understood to some degree by the installation security administrators, as they are the ones primarily dealing with data set security requirements. The extra element is any requirement for fault visibility between the groups.

Without using the XFACILIT resource class to control history file fault entry access, the security of PDS(E) history files is the data set security in your system. For users' jobs to record faults in the fault history file, they need write (update) access to the history file that the job uses. All users in the same group thus have read and delete access to the faults in a history file set up for the group.

## Setting up views

It is generally better to maintain separate fault history files for different user groups, providing the write (update) access only to the history file their jobs use. However, this approach makes it harder for a user to look at the faults across the environment if they are in many different history files. To solve this problem, create a "view" to let a user see a composite view of many history files in the one Fault Analyzer ISPF display.

---

## Managing history files across MVS systems without shared DASD

Views can be used for easy access to fault entries in more than one history file. However, this easy access is only possible when all history files in a view are accessible from the same MVS system.

In the following, two separate solutions are provided for sending fault entries from one system, on which the fault occurred, to another system for analysis:

- The first is an automated implementation, which requires no manual intervention. This implementation is described in "Automated implementation."
- The second is an implementation which can be used to transmit fault entries "on demand" during interactive reanalysis. This implementation is described in "On demand implementation" on page 270.

### Automated implementation

Given two MVS systems, 'A' and 'B', which do not share DASD, fault entries that are created on MVS system 'A' can be copied automatically to MVS system 'B' for viewing or reanalysis, by using the Fault Analyzer ISPF interface. (If the systems share DASD, they can simply use a common history file.) The entries are copied by using a Notification user exit on system 'A', which submits a TSO batch job to transmit fault entries as PDS members to a dedicated user ID on system 'B'. (See Chapter 31, "Customizing Fault Analyzer by using user exits," on page 359 for information about user exits in general and "Notification user exit" on page 397 for information about the Notification user exit specifically.) On system 'B', a continually running batch TSO job receives fault entries into a staging data set. It then calls the IDIUTIL batch utility to import the fault entry into a local history file. (See Chapter 27, "Managing history files (IDIUTIL utility)," on page 337 for information about the IDIUTIL batch utility.) Figure 132 on page 267 shows this concept.

## Managing history files across MVS systems without shared DASD

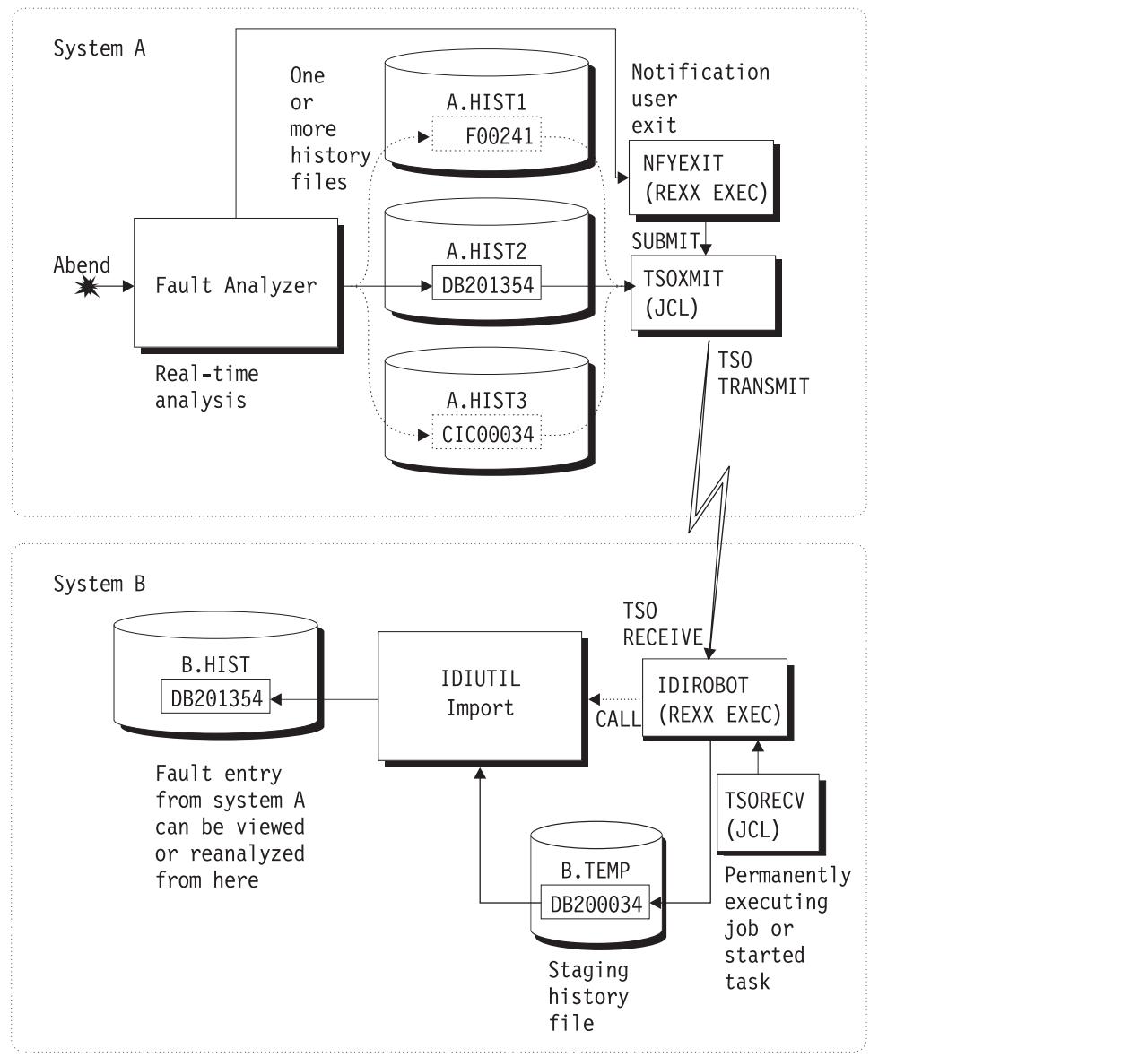


Figure 132. Fault entry import by way of TSO XMIT/RECEIVE

Figure 133 on page 268 shows the sample Notification REXX user exit, NFYEXIT. It is available in softcopy format as member IDISXNFY in data set IDI.SIDISAM1.

## Managing history files across MVS systems without shared DASD

```
nodeid = 'MVSB'          /* <-- verify/change */    1
userid = 'IDIROBOT'      /* <-- verify/change */    2
jobcard = '//NOTIFY JOB MSGCLASS=Z' /* <-- verify/change */    3
/********************************************************************* 4
/* #Optionally, add checks here for selective transmission of fault */
/* entries that only match a certain criteris. */
/* For example: */
/* If ENV.USER_ID ~= "FRED" then exit 0 */
/* If ENV.USER_IDIHOST ~= "MY.HISTFILE" the exit 0 */
/********************************************************************/
queue jobcard
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD SYSOUT=*" 
queue "//SYSTSIN DD *"
/* Split TSO XMIT command over two data records that must be
   padded with blanks to 80 bytes */
rec = "XMIT" nodeid".userid "DA('ENV.IDIHIST') -"
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
rec = "MEMBER("ENVFAULT_ID") NONOTIFY"
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
queue '/*'
/* 'Submit' the stacked TSO batch job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
address mvs "EXECIO" n "DISK DD1 (FINIS"
"IDIFREE DD(DD1)"
say 'Fault entry' ENVFAULT_ID 'sent to' nodeid'.userid
exit 0
```

Figure 133. Sample Notification user exit (NFYEXIT) to submit batch TSO XMIT job

### Notes<sup>®</sup>:

- 1** The 'nodeid' variable is used to designate the target system to which the fault entry is sent.
- 2** The 'userid' variable is the user ID for which fault entries are received on the target system. This user ID should be used solely to receive fault entries.
- 3** Ensure that the job card adheres to local standards.
- 4** You can add checks here to see whether a fault is eligible to be sent to another system. The example shows how the user ID or history file name can be used, but any fields in the ENV or NFY data areas might be checked.

The exit in Figure 133 can be called for any faults that occur on system 'A'. To facilitate this exit, add these options to the IDICNF00 config member, where *exec.lib* is your REXX EXEC PDS(E) data set:

```
DataSets(IDIEXEC(exec.lib))
Exits(NOTIFY(REXX(NFYEXIT)))
```

Figure 134 on page 269 shows the sample batch TSO receive REXX EXEC, IDIROBOT. It is available in softcopy format as member IDISROBT in data set IDI.SIDISAM1. This sample exec receives files for the IDIROBOT user into a staging history file from where they are imported into a local history file by using the IDIUTIL batch utility.

```

histfile = 'B.HIST'          /* <--- verify/change */ 4
temphist = 'B.TEMP'          /* <--- verify/change */ 5
seconds = '60'               /* <--- verify/change */ 6
address tso
x = prompt('on')
x = outrap('var.',10,'noconcat')
do forever
/* Obtain information about transmitted data on the JES output queue */
if queued() = 0 then queue 'end'
'receive'
input = 'N'

/* Examine the output from the 'dummy' receive command.
The following variables are initialized:
  dsn      - the 'sending' history file name
  fromid   - the user ID performing the TSO XMIT
  node     - the JES node from which the fault entry was sent
  faultid  - the fault ID (member name) */
do i = 1 to var.0
parse var var.i msgno t1 t2 t3 t4 t5 t6
if msgno = 'INMR901I' then do
  dsn = t2
  fromid = t4
  node = t6
end
else if msgno = 'INMR902I' then do
  faultid = t2
  input = 'Y'
  leave
end
end

/* Perform actual receive to the staging history file followed by an
IDIUTIL batch utility import if there is data available */
if input = 'Y' then do
/* The target history file in the 'histfile' variable could be
determined here based on any of the initialized variables above.
This sample EXEC uses a single history file only. */ 8
  say 'Receiving' dsn'('faultid') from' node'.fromid
  queue "da('tempHist')"
  queue 'end'
'receive'

/* Perform IDIUTIL batch utility import */
parm = "'import(histfile,tempHist)(faultid))'"'
address tso "call *(idiutil)" parm
say 'Import rc =' RC
end
else do
/* Sleep for 60 seconds before attempting to receive again */
  address tso "call *(idisleep) '\"seconds\""
end
end

```

---

*Figure 134. Sample TSO receive REXX exec (IDIROBOT)*

Notes:

- 4 This item is the name of the target history file in which the received fault entries are placed. To select the history file that is based on where the fault originally occurred, see 8.
- 5 This item is a staging history file that is used for the TSO receive

## Managing history files across MVS systems without shared DASD

command and from which fault entries are imported into the target history file and then deleted. The data set must be preallocated with attributes appropriate for a history file (PDS or PDSE, LRECL=10000, RECFM=VB) and with sufficient space to contain a single fault entry. To use the automatic space reclamation capabilities, make this data set a PDSE.

- 6 See 2 .
- 7 The IDIROBOT exec enters a WAIT state to preserve resources between checking for fault entries to be received. The time interval in number of seconds between receiving fault entries can be specified here. All fault entries on the JES output queue for the chosen user ID are received, then the IDIROBOT exec enters the WAIT.
- 8 The sample exec uses only a single target history file for all received fault entries. It is possible to assign a target history file that is based on one of these items:
  - The original history file name (in variable 'dsn').
  - The sending user ID (in variable 'fromid').
  - The node ID from where it was sent (in variable 'node'),
  - The fault ID itself (in variable 'faultid').

Figure 135 shows a sample batch TSO job to execute the IDIROBOT exec. It is available in softcopy format as member IDISTSOB in data set IDI.SIDISAM1.

---

```
//IDISTSOB JOB <job card parameters>
//TSOBATCH EXEC PGM=IKJEFT01
//SYSEXEC DD DISP=SHR,DSN=exec.lib
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSTSIN DD *
      IDIROBOT
/*

```

---

Figure 135. Sample TSO batch job to execute IDIROBOT exec (IDISTSOB)

It is important to ensure that the user ID under which the IDIROBOT exec is running (in this example, the submitter of the IDISTSOB job) has update access to both the staging history file and all history files used as targets for imports.

Because the IDIROBOT exec never exits, the IDISTSOB job executes indefinitely. However, the exec causes the job to enter a WAIT state between attempts to receive incoming data to prevent using unnecessary resources. To end the job, use the MVS CANCEL command during a period of inactivity. Alternatively, the exec could be made to recognize a special file that if sent to the selected user ID could trigger the exit to terminate.

A started task could be defined instead to execute this JCL in order to prevent tying up a JES initiator.

## On demand implementation

Instead of using a Notification user exit to automatically transmit all fault entries that match a certain criteria to another system for analysis, as described in “Automated implementation” on page 266, this implementation uses a Formatting user exit to perform the same function, but only as required from within the interactive reanalysis report.

## Managing history files across MVS systems without shared DASD

Figure 136 shows the sample Formatting REXX user exit, IDIXMIT. It is available in softcopy format as member IDIXMIT in data set IDI.SIDISAM1.

```
nodeid = 'MVS'                                /* <--- verify/change */
userid = 'IDIROBOT'                            /* <--- verify/change */
jobcard = '//NOTIFY JOB MSGCLASS=Z' /* <--- verify/change */
queue jobcard
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD SYSOUT=**"
queue "//SYSTSIN DD *"
/* Split TSO XMIT command over two data records that must be padded */
/* with blanks to 80 bytes. */
rec = "XMIT" nodeid"."userid "DA('ENV.IDIHIST') -"
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
rec = "MEMBER('ENVFAULT_ID') NONOTIFY"
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
queue '/*'
/* 'Submit' the stacked TSO batch job. */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
address mvs "EXECIO" n "DISKW DD1 (FINIS"
"IDIFREE DD(DD1)"
/********************************************************************* 1
/* #Optionally, update the user title field to show that the fault */
/* has been sent. */
/* For example: */
/* ENV.USER_TITLE = "Sent to" nodeid "on" DATE()." */
/********************************************************************* 2
/********************************************************************* 3
/* #Optionally, tell the user that the request has been processed */
/* by uncommenting the following IDIWRITE calls: */
/*"IDIWRITE '<p>History file" ENV.IDIHIST "fault ID" ENVFAULT_ID, */
/*"sent to" nodeid"."userid"." */
/*"IDIWRITE '<p>Press PF3 to return to the interactive reanalysis", */
/*"report." */
/********************************************************************* 4
/********************************************************************* 5
```

Figure 136. Sample Formatting user exit (IDIXMIT) to submit batch TSO XMIT job

Notes:

- 1** The 'nodeid' variable is used to designate the target system to which the fault entry is sent.

This target system might be an automated receiver (as discussed in “Automated implementation” on page 266, or any other node at which the receive and import is done manually.
- 2** The 'userid' variable is the user ID for which fault entries are received on the target system.

This ID might be an automated receiver (as discussed in “Automated implementation” on page 266, or any other user ID for which the receive and import is done manually.
- 3** Ensure that the job card adheres to local standards.
- 4** Uncommenting this code provides an optional method for identifying the fault entry as having been sent by providing the date on which this sending occurred, and the system to which it was sent, in the user title field.

## Managing history files across MVS systems without shared DASD

- 5** If you would like the user to see a display with information about what the issued command has performed, then uncomment the code here.

The receiving end of the manually transmitted fault entries might be the same automated received as discussed in “Automated implementation” on page 266, or it might be any user ID and node for which the receive, and subsequent import, is done manually.

### Using the on demand implementation

While analyzing a fault entry, enter the command

```
EXEC IDIXMIT
```

on the command line.

The IDIXMIT name can be any other name you have called your copy of the IDIXMIT sample exec.

---

## Sharing of history files across a sysplex

Fault Analyzer version 8.1 introduced the use of XCF messaging from IDIS subsystems to efficiently share PDSE history files across all MVS images in a sysplex. Previous contentions that would occur on the \$\$INDEX member of a history file during high activity are eliminated by the caching and XCF messaging between the IDIS subsystems in the sysplex. Combining this method with the PDSESHARING(EXTENDED) option in the IGDSMSxx member of SYS1.PARMLIB, it allows concurrent writing and reading of individual fault entries from all MVS images in a sysplex without contention. This change removes a previous recommendation not to share the writing to a history file from multiple MVS images in a sysplex.

As opposed to PDSE history files, sharing of PDS history files across a sysplex can cause undesired contention if high activity to the same history file occurs concurrently from multiple MVS images, since the access method requires that the entire data set is enqueued upon for any updates. PDSE data sets, on the other hand, only require serialization at the member level and are therefore a better choice.

Some environments have previously been set up with the unusual condition of not sharing master catalogs (or a user catalog) between all MVS images in a sysplex, and thus permitting individual MVS images to use the same data set name cataloged to a different DASD volume. Do not use this setup with Fault Analyzer history files, as the common data set name causes unnecessary contention through the IDIS subsystem and ENQ mechanisms between the MVS images. If different history files are desired between MVS images, then the use of the &SYSCLONE substitution variable in the history file data set name should be considered as the best alternative, since the unique naming then removes the common name ENQ contention and provides a more logical and practical management.

An alternative to using shared history files is to use the Fault Analyzer ISPF interface “View” facility, which allows multiple history files to be viewed simultaneously. For details of this facility, see “Setting up views” on page 263.

If sharing history files between multiple MVS images, the following must be observed:

- Systems sharing a PDSE must be members of the same sysplex and it is recommended that all systems are running with PDSESHARING(EXTENDED) in

the IGDSMSxx member of SYS1.PARMLIB. (Note that the first sysplex member to IPL determines the sharing mode used and forces all subsequent members that join the sysplex to run in that mode, whether EXTENDED or NORMAL.)

- XCF must be active.
- GRS (or a functionally equivalent subsystem) must be running. PDSE serialization is managed using resource major names SYSZIGW0 and SYSZIGW1. There is no need to make changes to GRS RNL lists, except if a serialization product other than GRS is used.
- See “Sharing of history files across a sysplex with mixed levels of Fault Analyzer” on page 216 for information about compatibility if sharing history files between sysplex images with a mix of Fault Analyzer versions or maintenance levels installed.

It is possible for OPEN errors to occur against PDSE data sets that are shared across a sysplex, irrespective of the above considerations. This situation might happen as a result of a cross-system sharing conflict, typically resulting in message IEC143I RC 213-70.

For more information about sharing of PDSE data sets in a sysplex, see chapter “PDSE Sharing and serialization” in the Redbooks publication “Partitioned Data Set Extended Usage Guide” (a copy can be downloaded from [www.redbooks.ibm.com](http://www.redbooks.ibm.com)) and the “Sharing PDSEs” section in chapter 28 “Processing a PDSE” in the “DFSMS: Using Data Sets” manual.

If an installation is required to use PDSESHARING(NORMAL), then the IDIS subsystem is also required, as this provides history file access management aimed at reducing cross-system sharing conflicts. However, if a history file is accessed outside of Fault Analyzer, then message IEC143I RC 213-70 might still occur.

No special Fault Analyzer options are required to use sysplex-shared fault history files.

It is recommended that the use of the IDIS subsystem PARM='UPDINDEX' option, or the absence thereof, is the same for all IDIS subsystems in the sysplex. For details about this option, see “Caching of history file \$\$INDEX data” on page 240.

---

## Managing history file fault entry access

It is possible to manage access to history files using security server data set profiles only, for example if a given history file is only used within a single department where all users are equal with regards to access authority. In this case, all users typically have at least UPDATE access to the history file and are thus able to create new fault entries, as well as view, modify, or delete any existing fault entries, regardless of who created them.

If more control over access to individual fault entries is required, then Fault Analyzer supports the use of the XFACILIT resource class as described in the following.

**Note:** Because XFACILIT access is associated with some degree of processing overhead, you might want to use history files with normal data set profile access for performance critical applications, such as CICS or IMS.

### Using the XFACILIT resource class for history file fault entries

If the required access to a history file fault entry is not already available via normal data set profiles, then the XFACILIT resource class can be used to provide the access via Fault Analyzer. To use the XFACILIT process for a history file, restrict the access available to the user via normal data set profiles, and then enable the access via the Fault Analyzer checked XFACILIT profiles. It is not possible to restrict Fault Analyzer access using the XFACILIT resource class beyond what the normal data set profiles allow, which is why the normal history file access should be set to UACC(NONE) for this set-up.

The design is to provide permission via XFACILIT, not restriction. For example, if a user cannot even browse a history file using ISPF 3.4, they could still be permitted to use the history file fault entries via Fault Analyzer with the XFACILIT profiles.

There are two XFACILIT class profile names used by Fault Analyzer—these are:

*IDIHIST\_GROUP\_DSN.group.hist-dsn*  
*IDIHIST\_USERID\_DSN.userid.hist-dsn*

where:

*group* and *userid*

are the security server default GROUP and USER ID that are associated with the current user (in case of creating a new fault entry), or the security server GROUP and USER ID that are associated with a fault entry (in the case of viewing, updating, or deleting an existing fault entry).

The security server GROUP and USER ID that are associated with a fault entry are those that were current for the user who initially created the fault entry.

*hist-dsn*

The name of the history file in which the fault entry resides.

Examples of XFACILIT class profile names could be:

*IDIHIST\_GROUP\_DSN.PAYROLL.IDI.HIST*  
*IDIHIST\_USERID\_DSN.USER01.SYSA.PROD.HIST*

If access is not available by normal data set profiles, then Fault Analyzer checks for access to a given history file fault entry using both of the above XFACILIT class profile names, that is, using group as well as user ID. If either is found to provide the access required, then the history file action is performed.

The XFACILIT class profile access levels used by Fault Analyzer translate to Fault Analyzer actions as follows:

*Table 6. Fault Analyzer XFACILIT profile access levels*

Fault Analyzer action	Required XFACILIT profile access level
Read.	READ
For example, viewing the saved report or performing reanalysis.	UPDATE or CONTROL

*Table 6. Fault Analyzer XFACILIT profile access levels (continued)*

Fault Analyzer action	Required XFACILIT profile access level
Delete.  This action applies to explicit deletions using the D (or DD) line command only.	ALTER

### **XFACILIT implementation example 1**

Given a history file that is named IDI.COMMON.HIST, then by preventing general access to the data set using, for example:

```
ADDSD 'IDI.COMMON.HIST' UACC(NONE)
```

and defining the following XFACILIT profile and access (repeated for each instance of *group*):

```
RDEFINE XFACILIT IDIHIST_GROUP_DSN.group.IDI.COMMON.HIST XFACILIT UACC(NONE)
PERMIT IDIHIST_GROUP_DSN.group.IDI.COMMON.HIST XFACILIT CLASS(XFACILIT) ID(group) ACCESS(UPDATE)9
```

then the resulting access to the IDI.COMMON.HIST history file is such that:

- Fault entries are accessible through Fault Analyzer only.
- Fault entries can be viewed or reanalyzed by:
  - The users who created the fault entries
  - Anyone else who is either a member of the same group, or who was granted explicit access to the XFACILIT profile containing the fault entry creator's default group ID

### **XFACILIT implementation example 2: Using global access table**

To use a history file TEST.ZZ.HISTORY.DEFAULT and have fault entry access protected by group ID and user ID, first prevent general access to the data set with:

```
ADDSD 'TEST.ZZ.HISTORY.**' UACC(NONE)
```

Then, to allow Fault Analyzer to grant access to the individual fault entry members in the PDS(E) data set, based on group ID and user ID, set up the XFACILIT class using the global access table:

```
SETOPTS GLOBAL(XFACILIT)
RDEFINE GLOBAL XFACILIT (<-- not required if already defined)
RALTER GLOBAL XFACILIT ADDMEM(IDIHIST_GROUP_DSN.&RACGPID.TEST.ZZ.HISTORY.**/ALTER)
RALTER GLOBAL XFACILIT ADDMEM(IDIHIST_USERID_DSN.&RACUID.TEST.ZZ.HISTORY.**/ALTER)
SETOPTS GLOBAL(XFACILIT) REFRESH
```

The global access table allows &RACUID and &RACGPID and so reduce the administrative effort.

#### **Note:**

1. For RACF, the &RACUID and &RACGPID only works for profiles listed in the global access table.
2. It is a RACF requirement, given how Fault Analyzer determines access authorization, that any XFACILIT profiles in the global access table are backed by a matching real XFACILIT profile. For example, add the following real XFACILIT profiles in order to enable the global access table profiles used here:

---

9. Additional groups or users can be given access to the XFACILIT string as appropriate.

## Managing history file fault entry access

```
PROFILE NOPREF
RDEFINE XFACILIT IDIHIST_GROUP_DSN.*.TEST.ZZ.HISTORY.** UACC(NONE)
RDEFINE XFACILIT IDIHIST_USERID_DSN.*.TEST.ZZ.HISTORY.** UACC(NONE)
SETR REFR RACLIST(XFACILIT)
```

The above XFACILIT definitions cover all history file data set names starting with the TEST.ZZ.HISTORY qualifiers, for example:

```
TEST.ZZ.HISTORY.DEFAULT
TEST.ZZ.HISTORY.PAYROLL
TEST.ZZ.HISTORY.CICS.SYS01
```

### XFACILIT implementation example 3: Using ACF2

OEM security servers need the commands converted to the specific product, such as the following ACF2 commands. The requirement is that the SAF RACROUTE requests give the equivalent return information for the OEM product, as would happen with RACF. The following is an example for ACF2.

```
$KEY(TEST)
ZZ.HISTORY.- UID(<string for MVS support>) READ(A) WRITE(A) ALLOC(A) EXEC(A)
ZZ.HISTORY.- UID(*)
```

Here, the TEST.ZZ.HISTORY.\* history files can only be directly accessed by the users in the <string for MVS support> list.

```
$KEY(IDIHIST_USERID_DSN) TYPE(XFC) or $KEY(IDIHIST_GROUP_DSN) TYPE(XFC)
-.TEST.ZZ.HISTORY.- UID(<string for MVS support>) ALLOW
-.TEST.ZZ.HISTORY.- UID(*) SERVICE(READ,UPDATE) ALLOW
```

TYPE(XFC) is the default for XFACILIT class. This setup only allows general users READ and UPDATE access, only MVS support can do explicit delete of fault entries. Automatic deletion is still done by Fault Analyzer according to the SetMaxFaultEntries/SetMinFaultEntries setting of each history file.

### Using the IDIXFXIT user exit

If access was not granted via the IDIHIST\_GROUP\_DSN.*group.hist-dsn* or IDIHIST\_USERID\_DSN.*userid.hist-dsn* XFACILIT profiles, then the optional IDIXFXIT user exit is called. If the IDIXFXIT exit is available, then it might ultimately grant the fault entry access. This approach might be useful if an installation has previously employed a different access authorization scheme and wants to continue using this instead of, or in conjunction with, the XFACILIT profiles.

The IDIXFXIT user exit must reside in an APF-authorized library and be available as a load module in LINKLIST with the name IDIXFXIT. It is loaded using the Language Environment CEELOAD service, and if LE conforming, should not contain a C or PL/I main() function. The IDIXFXIT user exit must be link-edited with the NORENT option.

**Entry specifications:** On entry to IDIXFXIT, the contents of registers are:

#### Register

#### Contents

- 1 Address of input parameter list (see below).
- 13 Address of 72-byte register save area.
- 14 Return address.
- 15 Entry point address of IDIXFXIT.

*Input parameter list:* The address of the following parameter list is passed to the IDIXFXIT user exit in R1. All parameters are provided as C-style null-terminated character strings, that is with the value followed by a byte containing X'00'.

*Table 7. IDIXFXIT input parameters*

Parameter	Number of bytes	Description
Parameter 1	Varying	The level of access required to the fault entry as one of the following: Read Update Delete
Parameter 2	Varying	Security server user ID of IDIXFXIT user exit caller <sup>1</sup> .
Parameter 3	Varying	Security server default group ID of IDIXFXIT user exit caller <sup>1</sup> .
Parameter 4	Varying	The name of the job that created the fault entry.
Parameter 5	Varying	The history file data set name containing the fault entry.
Parameter 6	Varying	The fault entry HD segment data area.  Mapping of the HD segment data area is provided in data set IDI.SIDISAM1 member IDISXPLA (Assembler), IDISXPLC (C), IDISXPLB (COBOL) and IDISXPLP (PL/I).

**Note:**

1. Obtain the fault entry creator security server user and default group ID information from the HD segment data area pointed to by parameter 6.

**Return specifications:** On return from IDIXFXIT, the contents of registers must be as follows:

**Register**

**Contents**

- |             |   |          |                     |          |                      |          |                        |          |                        |
|-------------|---|----------|---------------------|----------|----------------------|----------|------------------------|----------|------------------------|
| <b>0-1</b>  | Undefined.  |          |                     |          |                      |          |                        |          |                        |
| <b>2-14</b> | Unchanged.  |          |                     |          |                      |          |                        |          |                        |
| <b>15</b>   | Return code: <table> <tbody> <tr> <td><b>0</b></td> <td>Access not granted.</td> </tr> <tr> <td><b>1</b></td> <td>Read access granted.</td> </tr> <tr> <td><b>2</b></td> <td>Update access granted.</td> </tr> <tr> <td><b>3</b></td> <td>Delete access granted.</td> </tr> </tbody> </table> | <b>0</b> | Access not granted. | <b>1</b> | Read access granted. | <b>2</b> | Update access granted. | <b>3</b> | Delete access granted. |
| <b>0</b>    | Access not granted.   |          |                     |          |                      |          |                        |          |                        |
| <b>1</b>    | Read access granted.  |          |                     |          |                      |          |                        |          |                        |
| <b>2</b>    | Update access granted.  |          |                     |          |                      |          |                        |          |                        |
| <b>3</b>    | Delete access granted.  |          |                     |          |                      |          |                        |          |                        |

**Example (C):** The following is an example of an IDIXFXIT exit that is written in C.

```
#include "IDISXPLC"
int IDIXFXIT(char *action, char *userid, char *group, char *jobnm,
             char *histDSN, DDIR_DATA_HD *pHD) {
    if (strcmp("MY.HIST", histDSN) == 0) return 2; /* Update access ok */
    else return 0;                                /* No access */
}
```

## **Managing history file fault entry access**

---

## Chapter 19. Setting and changing default options for the site

The creation of a parmlib member, IDICNF $xx$ , that contains Fault Analyzer options which override the product defaults and are available to all users of Fault Analyzer at your site, is explained in the following.

### Parmlib member IDICNF $xx$

Default options for the site are held in the parmlib member IDICNF $xx$ , where  $xx$  is one of these:

- 00
- A value matching the current MVS symbol, &SYSCLONE.

&SYSCLONE is a standard MVS system symbol that represents a unique 1 or 2-character system ID. Refer to *MVS Initialization and Tuning Reference* for information about the MVS &SYSCLONE symbol.

Using an IDICNF&SYSCLONE member name permits an installation to specify different options for each MVS image in a sysplex, while still using a common PARMLIB data set.

The initial search is for an IDICNF&SYSCLONE member name. If this member name is not found in any parmlib data set, then a search is made for an IDICNF00 member.

The IDICNF $xx$  member can be created in SYS1.PARMLIB, or any other data set that is part of the logical parmlib concatenation.

All data sets in the logical parmlib concatenation must be given universal READ access.

**Note:** If you do not want to place the IDICNF $xx$  member in the logical parmlib concatenation, a USERMOD can be applied to Fault Analyzer that permits the specification of an alternative partitioned data set name. See “Specifying an alternative parmlib data set for IDICNF $xx$ ” on page 258 for details.

If the IDICNF $xx$  member does not exist, then Fault Analyzer uses the product-supplied default options. In real time, message IDI0018W is issued.

This is an example of an IDICNF $xx$  parmlib member:

## Parmlib member IDICNFxx

---

```
/*-----*/
/* IBM Fault Analyzer Configuration          */
/*-----*/
Exclude(TYPE(TSU))           /* Exclude TSO users          */
Exclude(TYPE(STC) NAME(VTAM)) /* Exclude VTAM started task */
RetainDump(AUTO)              /* Automatic dump retention   */
/* Data sets where installation application compiler listings are kept */
DataSets(
  IDIHIST (IDI.HIST)          /* Fault History file        */
  IDILC  (MY.LISTING.C)        /* C compiler listings        */
  XY.LISTING2.C)             /* more C compiler listings   */
  IDILCOB (APP1.LISTING.COBOL) /* COBOL compiler listings    */
  MY.LISTING.COBOL)          /* COBOL compiler listing for IVP */
  IDILCCBO(APP2.LISTINGS)     /* OS/VS COBOL compiler listings */
  IDILPLI (MY.LISTING.PLI)    /* PL/I compiler listings      */
  IDILANGX(MY.IDILANGX)       /* LANGX files                */
  IDIADATA(MY.SYSADATA)       /* SYSADATA files             */
  IDIMAPS(IDI.SIDIMAPS)        /* Data area models           */
)
)
```

---

Figure 137. Sample IDICNFxx parmlib member

A sample IDICNFxx parmlib member (which might not be the same as the sample above) is available in the IDI.SIDISAM1 data set as member IDICNF00.

Note that only columns 1 - 71 of the IDICNFxx member are processed.

When installing Fault Analyzer you must review the DataSets option in IDICNF00:

- If you have installed Fault Analyzer with a high-level qualifier other than IDI then you must include the DataSets option with specification of data sets for these DDnames:  
IDIMAPS  
IDIDOC
  - If you have allocated your VSAM message and abend code explanation repository with a name other than IDI.IDIVSENU, then you must include the DataSets option with specification of the data set name for the IDIVSENU DDname.
  - If you have allocated your default history file with a name other than IDI.HIST, then you must include the DataSets option with specification of the data set name for the IDIHIST DDname.

For details on the DataSets option, see “DataSets” on page 457).

You can change the parmlib member whenever you need to. For example, you can change the parmlib member to exclude other types of jobs. However, you should adjust an option in the user options file if you only want to change an option for one job.

Various techniques for changing options for individual jobs are described in “Where to specify options” on page 452, “Batch reanalysis options” on page 95, and “Interactive reanalysis options” on page 103.

The individual options are explained in “Option descriptions” on page 455.

**Note:** If a user-options module is used, it might replace the IDICNFxx default options. For details, refer to “User-options module IDICNFUM” on page 453.

The next sections briefly describe the function of some of the other Fault Analyzer options that you might want to consider specifying in the IDICNFxx parmlib member. However, if you are installing Fault Analyzer and merely want to reach a point at which the supplied IVP jobs can be run, then neither of these options are required.

In the remainder of this document, whenever references are made to the IDICNF00 parmlib member, it is implied that the name can be either IDICNF00, or IDICNF&SYSCLONE.

## Controlling which jobs are analyzed with Exclude processing

The term "work unit" is used in this section to refer to either a batch job, a started task, or a TSO user.

The Exclude and Include options ("Exclude/Include" on page 469) can be used to select which work unit abends you want analyzed.

If neither option is specified, the default is to include all work units.

It is possible to specify each of these options any number of times to achieve the desired inclusion or exclusion of specific work units. For example, all jobs can be excluded and then only certain jobs included, or if the opposite is desired, all jobs included (the default) and only some jobs excluded.

The selection process is as follows:

- The initial "state" of work unit selection is to include everything.
- Each specification of an Include or Exclude option is tested against the abending work unit:
  - If the criteria matches the abending work unit characteristics, then the current state is set to 'include' (if an Include option matched) or 'exclude' (if an Exclude option matched).
  - If the criteria does not match, then the state remains unchanged.
- If the final state after processing of all Include and Exclude options is 'exclude', then the abending work unit is excluded from analysis. In this case, you get no analysis report, and no fault entry is written to the history file.

### Note:

1. If you have excluded the real-time analysis, then you cannot later reanalyze, since there is no history file entry.
2. To make exclusion of analysis 'transparent' to the abending work unit, you might want to consider using the Quiet option. Otherwise, a message is issued to indicate that exclusion has occurred.

It follows from the above that the order in which Include and Exclude options are specified is significant. For example, if the following were specified, all batch jobs executing under the user ID FRED would be included for analysis, regardless of job name or execution class:

```
Exclude(TYPE(JOB) NAME(TEST*)) /* Exclude all batch jobs with names
                                 starting with TEST */
Exclude(CLASS(Z))           /* Exclude all batch jobs in class Z */
Include(TYPE(JOB) USERID(FRED)) /* Include batch jobs belonging to FRED */
```

## Controlling which jobs are analyzed with Exclude processing

However, if the last two options were reversed as shown, FRED's batch jobs would be excluded if they were executing in class Z:

```
Exclude(TYPE(JOB) NAME(TEST*)) /* Exclude all batch jobs with names
                                 starting with TEST */
Include(TYPE(JOB) USERID(FRED)) /* Include batch jobs belonging to FRED */
Exclude(CLASS(Z))             /* Exclude all batch jobs in class Z */
```

**Note:** Final exclude/include status can be controlled by an Analysis Control user exit (including dump registration) by setting the ENV.EXCLUDE value to "Y" or "N".

---

## Fast Exclude options processing

If the IDIS subsystem is started and the PARM='FASTEXCLUDE' option is in effect (this option is the default), then all Include and Exclude options specifications from the IDICNF00 parmlib member are cached in the IDIS subsystem. A Fault Analyzer invocation exit can obtain this information without the need for any I/O, and subsequently determine if a fault should be excluded from further processing prior to attaching the mainline IDIDA load module, where normal options processing is performed.

With the exception of CICS, only jobs or started tasks that do not include an IDIOPTS DDname are eligible for fast Exclude options processing. With CICS being a long-running system, Fault Analyzer periodically reads the Exclude/Include option specifications from the options data set, and caches these for subsequent use by the CICS invocation exits when an application abend occurs.

No IDITRACE information is provided for a fault that is excluded due to fast Exclude options processing.

If updating any Include or Exclude options in the IDICNF00 parmlib member, then the IDIS subsystem should be cycled to cause these options to be reread.

To disable fast Exclude options processing, specify the IDIS subsystem PARM='NOFASTEXCLUDE' option.

---

## Controlling report detail

The Detail option described at “Detail” on page 465 lets you specify the amount of detail you want in the analysis report.

This option does not affect the summary of the fault which is displayed on the operator console.

“The real-time analysis report” on page 18 describes the analysis report, and the effect that different Detail values have on the report.

You can change this value for a reanalysis.

---

## Limiting the size of minidumps

The MaxMinidumpPages option (“MaxMinidumpPages” on page 482) lets you limit the size of minidumps written to the history file. If the number of pages in the minidump exceed the limit in effect, then no minidump is written.

When choosing a minidump size limit, the space that is allocated to the fault history file should be considered. Each minidump page is 4 kilobytes.

---

## **Pointing to listings and REXX exec libraries**

The DataSets option (“DataSets” on page 457) lets you indicate where you have stored listings or side files. The specification is cumulative, so you can provide a global value, and then add to it with a further option for a particular job or fault reanalysis. Compiler listings or side files can also be specified via a user exit. For details, see “Compiler Listing Read user exit” on page 372.

Apart from listing and side-file data sets, the DataSets option can also be used to select other data sets, such as the history file and REXX exec user exit libraries.

---

## **Suppressing noncritical SYSLOG messages**

To prevent information and warning level messages from being written to the SYSLOG, use the Quiet option.

For information about the Quiet option, see “Quiet” on page 495.

---

## **Customizing the Fault Entry List display**

If an installation-wide change to the information that is shown on the Fault Entry List display is required, this change can be achieved by using the FAISPFopts(HistCols(...)) option to specify the particular columns of information to be displayed in the order that you want. This provides users with a common base from which they can make further changes if they want.

For information about the FAISPFopts option, see “FAISPFopts” on page 475.

---

## **Specifying the cultural environment**

The Locale option (“Locale” on page 480) lets you specify the locale to be used in order to enable cultural environment-dependent presentation.



---

## Chapter 20. Providing compiler listings or Fault Analyzer side files

When analyzing an abend, Fault Analyzer attempts to provide source line information obtained from compiler listings or side files. (For detailed information about the types of compiler listings or side files searched for, and the order of precedence, see “Locating compiler listings or side files” on page 290.)

If the relevant file cannot be located as a side file, but a compiler listing is found, then Fault Analyzer generates a side file from the listing and places it in a temporary data set, which is deleted once the analysis has completed.

If Fault Analyzer cannot find a listing either, then it is not able to provide source line detail, although it can still provide an analysis of the abend.

Even if a compiler listing or side file could not be found during real-time analysis, then one can be provided later during reanalysis of the history file fault entry. If necessary, create the compiler listing or side file by recompiling the abending program after the abend has occurred. However, in order for the compiler listing or side file to match the load module as at the time of the abend, and thus facilitate Fault Analyzer source level analysis, then it is important that the object deck from the recompilation remains unchanged.

Some compiler options, such as XREF, LIST, SOURCE and MAP, only change the information provided in the compiler listing. Other options, such as OPTIMIZE and SSRANGE, change the object code produced. When recompiling to add source support, all options should be left the same as the original compile, except for options that only affect the listing, such as LIST and XREF. Naturally, all input source, as well as the compiler version and maintenance level, should be the same.

For programs written in High Level Assembler, Fault Analyzer does not use the assembly listing but instead the SYSADATA file that is produced when specifying the assembler ADATA option. At the time of assemble, this data set is referenced by the SYSADATA DDname, but during Fault Analyzer processing it is read back via the IDIADATA DDname.

You have the choice of storing either listings or side files. However, side files are preferable for two reasons:

1. They use less disk space.
2. Fault Analyzer has to convert listings. Using side files reduces the total analysis time.

If you have set up PDS(E) data sets for compiler listings or side files, then you should make sure that they are populated with the programs likely to need analysis. Alternatively, compiler listings can be stored in sequential data sets.

Fault Analyzer searches all data sets specified for a given DDname separately and choose the best matching compiler listing or side file for a program. Thus, compiler listing or side file data sets containing different versions of the same program, for example a development version,a test version, and a production version, can all be provided simultaneously.

For information about the naming of compiler listings or side files, see “Naming compiler listings or side files” on page 289.

---

## COBOL Report Writer Precompiler

If you are using the COBOL Report Writer Precompiler (program number 5798-DYR), it is important that you run it as a stand-alone precompiler as opposed to invoking it via the COBOL compiler EXIT option. Otherwise, information that is required by Fault Analyzer to identify the point of failure source code statement might be missing from the compiler listing.

Symptoms that you might experience if using the COBOL Report Writer Precompiler as a COBOL compiler exit are:

- Return code 3114 from IDILANGX if trying to convert the COBOL compiler listing file to a side file.
- The following messages are issued during fault analysis:
  - IDISF8100S COBOL LISTING file contains NO recognized records
  - IDISF8132S Input or Output file format invalid
- Failure to determine point of failure source line.

---

## Required compiler options for IDILANGX

To create LANGX side files, the IPVLANGX utility is used. The IPVLANGX utility is provided with Problem Determination Tools Common Component. IDILANGX is an alias of IPVLANGX. For details about running the IPVLANGX utility, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*, chapter “IPVLANGX compiler listing to side file conversion utility”.

The following are the compiler options needed to produce listings or side files suitable for Fault Analyzer. If compiler-generated TEST(SEPARATE) SYSDEBUG side files are used, then these options do not matter.

C:

AGGREGATE  
LIST  
NOOFFSET  
NOOPT (Note 1)  
SOURCE  
XREF  
NOIPA

C++:

ATTRIBUTE (Note 4)  
LIST  
LONGNAME  
NOOFFSET  
NOOPT (Note 1)  
SOURCE  
XREF  
NOIPA

OS/VS COBOL:

DMAP  
NOCLIST  
NOLST  
NOOPT (Note 1)

PMAP  
SOURCE  
VERB  
XREF

**COBOL compilers other than OS/VS COBOL:**

LIST,NOOFFSET (Note 2)  
NOOPT (Note 1)  
MAP  
SOURCE  
XREF(SHORT) (Note 3)

**Enterprise PL/I:**

AGGREGATE  
ATTRIBUTES(FULL)  
NOBLKOFF  
LIST (Note 5)  
MAP  
NEST  
NONUMBER  
OFFSET  
NOOPT (Note 1)  
OPTIONS  
SOURCE  
STMT  
XREF(FULL)

**PL/I compilers other than Enterprise PL/I:**

AGGREGATE  
ATTRIBUTES(FULL)  
ESD  
LIST (Note 5)  
MAP  
NEST  
NOOPT (Note 1)  
OPTIONS  
SOURCE  
STMT  
XREF(FULL)

**Assembler:**

ADATA

**Notes:**

- 1      Although NOOPT is recommended, the use of OPTIMIZE is allowed (including OPT(1) or OPT(2) for C), in which case the compiler merges and rearranges statement numbers in the compiled code. The Fault Analyzer analysis is limited to what can be determined from the optimized compiler listing, which can vary from having no effect on the Fault Analyzer report, to inaccurate identification of the source line that failed. The source line number is usually close, but not necessarily accurate with OPTIMIZE. It is dependent on the compiler's rearrangement or elimination of source statements during its optimization processing. Data field values cannot be shown if storage has not been assigned, that is, if the value is in a register only.
- 2      Although LIST and NOOFFSET are recommended, the use of NOLIST and

## Required compiler options for IDILANGX

OFFSET is allowed, in which case Fault Analyzer is not able to warn the user if the compiler listing is not a good match with what is in storage.

- 3 XREF(SHORT) is a minimum requirement; XREF(FULL) is permitted and has no detrimental effect.
- 4 ATTRIBUTE is a minimum requirement; ATTRIBUTE(FULL) is permitted and has no detrimental effect.
- 5 Although the use of NOLIST and OFFSET is allowed, Fault Analyzer is not able to warn the user if the compiler listing is not a good match with what is in storage. Also, it might result in parameter variables being omitted from the analysis report.

LIST is required to correctly report Enterprise PL/I STATIC EXTERNAL variables and parameters.

## TEST option considerations

With all compilers, the use of the TEST option provides program information in addition to what is available from the side files.

If TEST(NONE,SYM,SEPARATE) is used when compiling a COBOL program, or TEST(STMT,SYM,NOHOOK,SEPARATE) is used when compiling an Enterprise PL/I program, then a SYSDEBUG file that is suitable for use by Fault Analyzer is written. (See “Locating compiler listings or side files” on page 290 for information about how the SYSDEBUG file is searched for by Fault Analyzer.) If the SYSDEBUG file is to be used instead of a compiler listing, or a LANGX side file created from a compiler listing, then it should be retained for use by Debug Tool for z/OS and Fault Analyzer.

The COBOL SYSDEBUG side file typically uses about 30% less DASD space than a Fault Analyzer IDILANGX side file.

The Enterprise PL/I SYSDEBUG side file is not a stand-alone debugging aid (unlike the COBOL equivalent, which is). The load module always contains the statement number table (which provides source line offsets), and also contains symbol information when the program has GET/PUT DATA statements.

## DEBUG option considerations

Use the DEBUG option with the XL C/C++ compilers to write DWARF debugging information. This debugging information can be written to an MVS data set or HFS file. Fault Analyzer locates DWARF files from information in the load module.

DWARF files for all the compile units in a load module can be combined, together with source, into an MDBG side file (the CDADBLD utility is used for this). Like DWARF, MDBG files can reside in an MVS data set or HFS file. MDBG files stored in MVS data sets are located via the IDISYSDB DD. MDBG side files in HFS are self-locating, and as such must adhere to the following:

- MDBG files should reside in the same HFS directory as the original DWARF files.
- The name of the MDBG file should be the same as the load module it was created from, and have a file extension of .mdbg.
- The MDBG file name and extension cannot consist of mixed case characters. The case of both the file name and extension are determined from the file name and extension of the original DWARF file for the compile unit (thus, it is possible for the file name to be upper case while the file extension is lower, and vice versa).

---

## Naming compiler listings or side files

Store compiler listings or side files in sequential data sets, or as members of PDS(E) data sets.

If stored in PDS(E) data sets, then the member name that should be used depends on the programming language. For details, see to the appropriate language section in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*, chapter "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products".

In addition to the information that is provided here, the following applies to the member names of Assembler side files:

- For Assembler single-CSECT programs (that is, one CSECT in each assembly), use the CSECT name or the load module name. Using the CSECT name is preferable to using the load module name, as there is less overhead in locating the member. Also, the load module name should not be used if the assembler CSECT is only a subroutine.
- For Assembler multi-CSECT programs (that is, more than one CSECT in each assembly), use the load module name.

If you store with any other name, then Fault Analyzer is unable to find the compiler listing or side file.

If compiler listings or side files are stored in sequential data sets, and the data set names follow a convention that permits the program name to be part of the data set name, then the specification of these data sets in the DataSets option can be done easily using variable substitution, as described in "Data set name substitution symbols" on page 460.

---

## Naming CSECTs for Fault Analyzer

To facilitate source code information, Fault Analyzer must be able to match CSECT names with the compiler listings or side files provided. For this to be possible, all CSECTs must be named. Whereas the names of CSECTs in programs written in most high-level languages are automatically assigned, special requirements apply to programs written in C or assembler, as explained in the following. Failure to follow these requirements prevents source code information from being determined for these types of programs.

For information about C program CSECT naming requirements, see the section "z/OS XL C and C++ programs" in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*, chapter "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products".

For information about assembler program CSECT naming requirements, see the section "Assembler programs" in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*, chapter "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products".

---

## Locating compiler listings or side files

Fault Analyzer chooses a matching compiler listing or side file from the first of the following possible sources:

1. TEST option data:

- If the program was compiled with the TEST option (other than TEST(SEPARATE) for COBOL or Enterprise PL/I), then a temporary Fault Analyzer side file is generated from debug information contained in the load module.
- If a COBOL or Enterprise PL/I program, and it was compiled with TEST(SEPARATE)<sup>10</sup>, then the associated SYSDEBUG side file data set name is obtained from debug information that was created by the compiler and placed in the load module.
- If a COBOL program compiled with TEST(SEPARATE), then the SYSDEBUG side file data set name is used as input to the optional COBOL IGZIUXB exit. This exit might return a different side file data set name.

**Note:** Under CICS, the batch form of EQAUEDAT is used as Fault Analyzer runs in an attached TCB without CICS command access.

2. The Debug Tool for z/OS EQAUEDAT exit (optional).

Refer to *Debug Tool for z/OS: Customization Guide* for details about the EQAUEDAT exit.

If a COBOL or Enterprise PL/I program compiled with TEST(SEPARATE), then the SYSDEBUG side file data set name from the load module is used as input to the EQAUEDAT exit.

No input side file name is provided when searching for IDILANGX or compiler listings.

This exit might return a different side file data set name.

**Note:** Under CICS, the batch form of EQAUEDAT is used as Fault Analyzer runs in an attached TCB without CICS command access.

3. If a COBOL or Enterprise PL/I program, then any SYSDEBUG data sets provided via the IDISYSDB DDname are searched. For XL C/C++ programs, MDBG data sets provided via the IDISYSDB DDname are searched. The data sets are searched individually, that is, they are not treated as a logical concatenation.

4. Fault Analyzer side files:

- Compiler Listing Read user exit called with request for Fault Analyzer side file.
- Fault Analyzer side file data sets provided via the IDILANGX DDname are searched. The data sets are searched individually, that is, they are not treated as a logical concatenation.

5. Compiler listings:

- Compiler Listing Read user exit called with request for compiler listing (or an Assembler SYSADATA file).
- Language-specific compiler listing data sets provided via the appropriate DDname are searched (for example, IDILCOB). The data sets are searched individually, that is, they are not treated as a logical concatenation.

---

10. See “TEST option considerations” on page 288 for recommended language-dependent suboptions.

The determination of whether a compiler listing or side file is satisfactory for use by Fault Analyzer to map a program depends on a number of things:

- **If the NOTEST compiler option is used**

When Fault Analyzer finds a compiler listing or LANGX side file for a program, a number of tests are performed, subject to the programming language used, to verify that the file matches the load module:

- **All languages**

From the Assembler listing section of any compiler output, Fault Analyzer extracts the last few assembler instructions and compares them against the load module, and these assembler instructions must be at the correct offset in the load module according to the listing. If any of these instructions are not found at the correct offset in the load module, then this check fails, and the compiler listing or side file is not used to provide source-level information.

- **COBOL-specific tests (excluding OS/VS COBOL)**

For COBOL, four more length values are extracted from the listing. These are the TGT length, WORKING-STORAGE length, number of Data Division statements, and number of Procedure Division statements from the following four lines found in the compiler listing:

```
TGT      WILL BE ALLOCATED FOR nnnnnnnn BYTES
WRK-STOR WILL BE ALLOCATED FOR nnnnnnnn BYTES
Data Division statements = nnnnnn
Procedure Division statements = nnnnnn
```

These four values are compared to those found in the load module, and if either do not match, then the compiler listing or side file is not used to provide source-level information.

- **If the TEST compiler option is used**

For COBOL SYSDEBUG files, the date and time comparisons are replaced with a 'signature' check, allowing an unchanged program to be recompiled. The SYSDEBUG file can still be used during interactive reanalysis if the signature check fails by following the compiler listing mismatch pop-up displays.

For COBOL programs, up to and including Enterprise COBOL V4, but excluding VS COBOL II, the compiler listing data set name is obtained from debug information in the load module, placed there by the compiler. In this case, the date and time in the listing are checked against the compile date and time in the load module. If the compiler listing file has been moved (or it is VS COBOL II), then a compiler listing can be specified by way of the IDILCOB DD, EQAUEDAT or the Compiler Listing Read user exit.

For Enterprise COBOL V5 programs compiled with TEST(NOSOURCE) or NOTEST(DWARF), no source information is included in the program object and no compiler listing data set name is available in the DWARF. However, compiler listings for these programs can be specified by way of the IDILCOB DD, EQAUEDAT or the Compiler Listing Read user exit.

For Enterprise PL/I SYSDEBUG files, the date and time comparisons are replaced with a set of 'integrity' checks, which determine whether the SYSDEBUG file can be used with the load module in storage. If any of the integrity checks fail, then the SYSDEBUG file cannot be used.

The reason why Fault Analyzer performs these checks prior to using the data is that an incorrect compiler listing or side file might produce very misleading report information.

If no satisfactory compiler listing or side file was found for a COBOL, PL/I, C/C++, or assembler program during interactive fault reanalysis, then Fault

## Locating compiler listings or side files

Analyzer prompts the user to optionally supply the location of a compiler listing, Fault Analyzer side file, or COBOL SYSDEBUG side file. For more details on this prompt, see “Prompting for compiler listing or side file” on page 166.

## IDITRACE information

By adding an IDITRACE DDname to your job, then Fault Analyzer can provide you with trace information that might help you understand why a particular compiler listing or side file was selected or rejected. For example:

```
//IDITRACE DD SYSOUT=*
```

(See “IDITRACE under CICS” on page 307 for an alternative method of activating this trace under CICS.)

The trace information is written to the IDITRACE DDname destination. An example of compiler listing or side file search trace information follows:

---

```
Listing/Side File search trace for IDISCB1
Execution program IDISCB1 compile date 2013/02/18 time 12:17:24
Rejected - DA.LISTING.COBOL(IDISCB1)
Failed -
    OPEN error, member not found.
Rejected - SWILKEN.$$TEMP$$.LISTINGS(IDISCB1) Built 2013/02/18 12:17:41
Failed -
    COBOL Working Storage length mismatch - load x'2A' listing x'32'
    COBOL Data Division Statements mismatch - load 6 listing 7
Passed -
    COBOL TGT lengths match load and listing both 148
    COBOL Procedure Division Statements match, both 10
    Object code length check passed.
Accepted - SWILKEN.IVPCB.LISTINGS(IDISCB1) Built 2013/02/18 12:17:24
Passed -
    COBOL TGT lengths match load and listing both 148
    COBOL Working Storage lengths match load and listing both x'2A'
    COBOL Data Division Statements match, both 6
    COBOL Procedure Division Statements match, both 10
    Object code length check passed.
```

---

Figure 138. Sample compiler listing/side file search trace

When performing interactive reanalysis, then IDITRACE information pertaining to the search for compiler listings or side files is available by selecting an option from the Compiler Listing Not Found display, without the need to allocate the IDITRACE DDname—for details, see “Prompting for compiler listing or side file” on page 166.

## Using the IDIXSFOR compiler listing/side file search and override exit

If available, then Fault Analyzer invokes the IDIXSFOR exit to allow programmatic specification of the location of compiler listing or side file data. The input parameter list is similar to that used by EQAUEDAT, with an extra FORCE parameter added to the end. The FORCE parameter can be used to tell Fault Analyzer to accept a side file, even if the last 12 instructions of the object code do not match. The FORCE parameter can also be used to request that no further side file processing should be performed for the current compile unit.

### IDIXSFOR invocation

The following describes the entry and return specifications for IDIXSFOR.

#### Entry specifications

On entry to IDIXSFOR, the contents of registers are:

##### Register

##### Contents

- 1** Address of input/output parameter list (see below).
- 13** Address of 72-byte register save area.
- 14** Return address.
- 15** Entry point address of IDIXSFOR.

**Input parameter list:** Register 1 must contain the address of a parameter list, consisting of a list of addresses (OS linkage). Each address in the parameter list points to a parameter.

*Table 8. IDIXSFOR input parameters*

Parm #	Length	Input/Output	Description																		
1	See parm #2	Input/Output	Side file data set name <b>Input</b> Fault Analyzer might provide an input data set name if the compile option was TEST(SEPARATE). <b>Output</b> Exit-provided new data set name/path to try. The length of this name must not exceed 1024 bytes. The member name must be included if a PDS(E).																		
2	4	Input/Output	Side file data set name (parm #1) length <b>Input</b> Length of input side file name. <b>Output</b> Length of new data set name or path.																		
3	4	Input	Language code <table> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>3</td><td>C source</td></tr> <tr> <td>4</td><td>COBOL</td></tr> <tr> <td>7</td><td>Assembler, OS/VS COBOL, non-LE VS COBOL II</td></tr> <tr> <td>10</td><td>PL/I (other than Enterprise PL/I)</td></tr> <tr> <td>11</td><td>Enterprise PL/I</td></tr> <tr> <td>35</td><td>C side file that is compiled with FORMAT(DWARF)</td></tr> <tr> <td>37</td><td>C MDBG side file</td></tr> <tr> <td>41</td><td>VS COBOL II (under LE)</td></tr> </tbody> </table>	Value	Meaning	3	C source	4	COBOL	7	Assembler, OS/VS COBOL, non-LE VS COBOL II	10	PL/I (other than Enterprise PL/I)	11	Enterprise PL/I	35	C side file that is compiled with FORMAT(DWARF)	37	C MDBG side file	41	VS COBOL II (under LE)
Value	Meaning																				
3	C source																				
4	COBOL																				
7	Assembler, OS/VS COBOL, non-LE VS COBOL II																				
10	PL/I (other than Enterprise PL/I)																				
11	Enterprise PL/I																				
35	C side file that is compiled with FORMAT(DWARF)																				
37	C MDBG side file																				
41	VS COBOL II (under LE)																				
4	See parm #5	Input	Compile unit name																		
5	4	Input	Compile unit name (parm #4) length																		
6	See parm #7	Input	Load module name																		

## Using the IDIXSFOR compiler listing/side file search and override exit

Table 8. IDIXSFOR input parameters (continued)

Parm #	Length	Input/Output	Description
7	4	Input	Load module name (parm #6) length
8	See parm #9	Input	Load library name
9	4	Input	Load library name (parm #8) length
10	4	Output	Force return option word
			Value      Meaning
			0      No forced override
			1      Force accept mismatch of last 12 object instructions
			-1      Ignore side file processing for this compile unit

### Return specifications

On return from IDIXSFOR, the contents of registers are:

Register	Contents
----------	----------

- 0-1      Undefined.
- 2-14     Unchanged.
- 15       Undefined.

### Example (Assembler)

The following is an example of an IDIXSFOR assembler exit.

```

TITLE 'IDIXSFOR HLASM EXAMPLE'
*
*      The map of input/output parameters
*
PARMLIST DSECT
DSETNM@ DS   A      Address of side file data set name
DSETLEN DS   A      Address of side file data set name length
LANGCODE DS   A      Address of language code
CUNM@   DS   A      Address of CU name
CUNMLEN DS   A      Address of CU name length
LDMDMN@ DS   A      Address of load module name
LDMDMNLN DS   A      Address of load module name length
LDLBMN@ DS   A      Address of load library name
LDLBMNLN DS   A      Address of load library name length
FORCE@  DS   A      Address of force return option word
*
*      Language Codes
*
LANCOBOL EQU    4      COBOL
LANVSCBL EQU   41     VS COBOL II (under LE)
LANPLI   EQU   10     PL/I (other than Enterprise PL/I)
LANPLIEN EQU   11     Enterprise PL/I
LANC    EQU    3      C source
LANCDWRF EQU   35     C side file compiled with FORMAT(DWARF)
LANCMDBG EQU   37     C MDBG side file
LANASSEM EQU    7      Assembler, OS/VS COBOL, Non-LE VS COBOL II
*
*      force options
*
FORCEL12 EQU   1      Force accept mismatch of last 12 object inst.
FORCEIGN EQU  -1      Ignore side file processing for this CU

```

## Using the IDIXSFOR compiler listing/side file search and override exit

```

*
*      Prologue
*
*      IDIXSFOR CEEENTRY AUTO=DSASIZ,          Amount of main memory to obtain      *
*                  PPA=PPA3,                   Program Prolog Area for this routine   *
*                  MAIN=NO,                   This program is a Subroutine           *
*                  NAB=NO,                   NO- not called from LE-enabled pgm    *
*                  PARMREG=R3,              R1 value is saved here                 *
*                  BASE=R11                Base register for executable code,
*                                         constants, and static variables
*
*      USING CEECAA,R12          Common Anchor Area addressability
*      USING CEDDSA,R13          Dynamic Storage Area addressability
*      USING PARMLIST,R3
*      L   R4,DSETNM@          Addr of side file data name address
*      L   R4,0(R4)             Addr of side file data name
*      L   R5,DSETLEN          Addr of side file data name length
*
*      L   R6,LANGCODE          Addr of language code
*      L   R6,0(R6)              Language code
*
*      L   R7,CUNM@            Addr of CU name address
*      L   R7,0(R7)             Addr of CU name
*      L   R8,CUNMLEN          Addr of CU name length addr
*
*      L   R9,LDMDMN@          Addr of load mod name address
*      L   R9,0(R9)             Addr of load mod name
*      L   R10,LDMDMNLN        Addr of load mod name length
*
*      L   R2,LDLBMN@          Addr of load library name address
*      L   R2,0(R2)             Addr of load library name
*      L   R0,LDLBMNLN          Addr of load library name length
*      C   R6,=A(LANCOBOL)     LANGCODE exit if not COBOL
*      BNE   EXIT
*
*      Typical processing will use the load library DSN and the
*      compile unit name to determine the side file DSN with the
*      member name normally being the compile unit name.
*
*      When the input DSETNM is provided for a TEST(,SEPARATE) compile,
*      the LDLBNM may show (via data set naming conventions) the
*      application has been promoted from test to production. This
*      would allow input DSETNM generated at compile time to be altered
*      according to the data set naming conventions to locate the new
*      (promoted) location of the side file.
*
NEXT   EQU   *
*      If input load dsn name is PROD.LOAD, then
*      tell FA to ignore last 12 instruction mismatch
*      and change the DSN to TEST.IDILCOB
*      CLC   =C'PROD.LOAD',0(R2)
*      BNE   EXIT
*      L   R1,FORCE@          force accept mismatch last 12 inst.
*      MVC   0(4,R1),=A(FORCEL12)
*      MVC   0(14,R4),=C'''TEST.IDILCOB('
*      MVC   14(8,R4),0(R7)    add member name = CU name
*      L   R6,0(,R8)           load CU name length
*      LA   R6,14(R4,R6)       point to end
*      MVC   0(2,R6),=C')'''  close bracket and quote
*      LA   R6,2(R6)           include in length
*      SR   R6,R4              subtract beginning
*      ST   R6,0(R5)           set new name length
*
EXIT   EQU   *
*
*      Epilogue
*
*      CEETERM RC=0
*      LTORG ,

```

## Using the IDIXSFOR compiler listing/side file search and override exit

```
*  
*  
*      Symbolic Register Definitions and Usage  
*  
R0    EQU  0          Work register  
R1    EQU  1          Parameter list address (upon entry)  
R2    EQU  2          Work register  
R3    EQU  3          Parameter list address (after CEEENTRY)  
R4    EQU  4          Work register  
R5    EQU  5          Work register  
R6    EQU  6          Work register  
R7    EQU  7          Work register  
R8    EQU  8          Work register  
R9    EQU  9          Work register  
R10   EQU  10         Work register  
R11   EQU  11         Base register for executable code  
R12   EQU  12         Common Anchor Area address  
R13   EQU  13         Save Area/Dynamic Storage Area address  
R14   EQU  14         Return point address  
R15   EQU  15         Entry point address  
*  
PPA3   CEEPPA ,       Program Prolog Area for this routine  
*  
*      Map the Dynamic Storage Area (DSA)  
*  
        CEEDSA ,       Map standard CEE DSA prologue  
*  
*      Local Automatic (Dynamic) Storage..  
*  
DSASIZ EQU  *-CEEDSA Length of DSA  
*  
*      Map the Common Anchor Area (CAA)  
*  
        CEECAA  
END   ,             of IDIXSFOR
```

---

## Compiler listings and side file attributes

Compiler listings and side files must be allocated using the following attributes:

DDname	Attributes
--------	------------

**IDIADATA**

Sequential data set or PDS(E), RECFM=VB, LRECL≥8188

**IDILC** Sequential data set or PDS(E) with either of the following attributes:

- RECFM=VB or VBA and LRECL≥137
- RECFM=FB or FBA and LRECL=133

**IDILCOB**

Sequential data set or PDS(E), RECFM=FBA, LRECL=133

**IDILCOBO**

Sequential data set or PDS(E), RECFM=FBA, LRECL=121

**IDISYSDB**

Sequential data set or PDS(E), RECFM=F or FB, 80≤LRECL≤1024

**IDILANGX**

Sequential data set or PDS(E), RECFM=VB, LRECL≥1562

**IDILPLI**

Sequential data set or PDS(E), RECFM=VBA, LRECL≥125

### IDILPLIE

Sequential data set or PDS(E), RECFM=VBA, LRECL $\geq$ 137

In order for Fault Analyzer to read the compiler listings or side files, they must not be allocated as temporary data sets (for example, using &&dsname-type data set names in your JCL).

For the purpose of conserving disk space, compiler listings can be stored in ISPF packed format. This storage is done by using the PACK ON option from within ISPF edit of the file. The ISPF packed format is not permitted for IDILANGX or IDIADATA data sets.

---

## Using the IDIRLOAD DDname for CSECT mapping

Since Fault Analyzer generally searches for compiler listings or side files using the uppercase eight-character CSECT name, it follows that resolving CSECT names is a pre-requisite to presenting source level information, such as the point-of-failure source line or statement.

During real-time processing, Fault Analyzer automatically invokes the MVS Binder program to perform the mapping of CSECTs in application load modules. CSECTs are usually not determined in non-application load modules, such as those belonging to the MVS operating system or Language Environment, for performance reasons.

Also, when performing MVS dump analysis using the ISPF interface "File->Analyze MVS Dump Data Set" action-bar pull-down menu option (for details, see "Action-bar pull-down menus" on page 61), then CSECT mapping is generally not available. This lack of availability might be due to the inability to determine the data set name from where a load module was originally loaded, or because the dump is analyzed on a different system to where it was written.

To enable CSECT mapping of non-application load modules, or any load modules in the case of MVS dump analysis, the user can pre-allocate a concatenation of load libraries to the IDIRLOAD DDname. This allocation of the IDIRLOAD DDname can, for example, be performed using the TSO ALLOCATE command prior to the dump analysis, or using the IDIALLOC REXX command from an Analysis Control user exit.

If a Analysis Control user exit is used, then there is also the option to use an alternative DDname, which must be identified via the ENV.IDIRLOAD\_DD field (for details, see "ENV - Common exit environment information" on page 513). This approach can be useful if performing multiple simultaneous Fault Analyzer dump analyses in separate ISPF split screen sessions.

If during reanalysis, Fault Analyzer does not have the link-edit information for a load module due to any of the reasons mentioned above, then it uses the IDIRLOAD concatenation to search for a module of the same name and uses IEWBIND to extract the CSECT name information. This approach facilitates potential IDILANGX source mapping for those CSECTs.

It is important that any load libraries that are allocated to the IDIRLOAD DDname match the load modules being mapped, otherwise incorrect CSECT and source line determination might result.

## Compiler listings and side file attributes

To facilitate source level analysis of PL/X programs, the user must provide matching Fault Analyzer side files, created by calling IDILANGX with PARM='mbr\_name (PLX)' and specifying both SYSADATA and SYSLOGIC input data.

## IDIDATST side file availability test utility

A utility program, IDIDATST, is provided to help determine the availability of matching source mapping side files for programs within load modules. Normally, IDIDATST would be used during an installation phase, if it had been decided to check if existing compile listings or side file rebuild processes had adequately produced usable side files, should abends occur in the program suite. It is a way of determining before an abend occurs, if matching compile listings or side files are available for a program.

The basic operation of IDIDATST, with no input parameters, is to scan all programs in load modules of the //STEPLIB DD first concatenation data set. Each program or CSECT is tested for an available matching side file from the corresponding IDICNFxx DataSets option list of side file data sets, and a search report is written to //IDIREPORT DD, if there is one in the job step. All of the side file DD statements for each language, such as //IDILCOB for COBOL, can be added to the IDIDATST job step, and they are searched first for the corresponding program language before searching any IDICNFxx options data sets.

A sample job for invoking the IDIDATST utility is as follows:

```
//IDIDATST JOB ...
//RUN      EXEC PGM=IDIDATST,PARM='parms'
//STEPLIB  DD   DISP=SHR,DSN=load_lib_dsn
//IDICSV   DD   SYSOUT=*
```

The optional PARM field that can be added to the EXEC PGM=IDIDATST JCL statement is in the form:

### Syntax

►—PARM='PGM=load\_module\_name'————►

where:

*load\_module\_name*

The name of a load module in the //STEPLIB DD first concatenation data set. If the load module name that is specified is shorter than 8 characters, then it is used as a filter to match against the same first characters of all load module names.

For example, specifying

PARM='PGM=KACBB092'

tests only the load module KACBB092, whereas specifying

PARM='PGM=KAC'

tests all load modules with names starting with KAC in the first STEPLIB data set.

A consolidated report is created from each run of IDIDATST, with a one line result for each program search, which is written to the //IDICSV DD. The IDICSV

output is in a Comma Separated Value format, which might be useful if importing the data into a spreadsheet program, or similar. For a copy of the same data in a column aligned format, which is more suited to editor review, add a //IDIFLAT DD to get the CSV data aligned with blanks.

The IDIDATST process for a STEPLIB remembers which programs or CSECTs have been tested, along with their compile date, and does not retest them if they occur in a subsequent load module.

IDIDATST cannot exactly replicate a side file test that would occur under an abend condition.

An optional //IDINOTST DD can be added to provide a blank-delimited list of prefixes for program names that are to be excluded from side file checking. For example:

```
//IDINOTST DD *
XYZ ERR1*
ABCD
/*
```

Appending an asterisk to the program name prefix is optional and does not alter the IDIDATST behavior.

---

## ISPF-packed compiler listings

ISPF provides the facility to PACK data sets. In this format, ISPF replaces any repeating characters with a sequence showing how many times the character is repeated. This facility can allow you to use direct access storage devices (DASD) more efficiently. When opening a compiler listing, Fault Analyzer detects if the listing is in PACKed format and unpacks it before use.

To pack a compiler listing under ISPF, one can issue the PACK primary command from within an EDIT session on the listing, and then save the listing.

Alternatively, the creation of packed compiler listings can be automated by inserting an extra step into your compile job. An example job is shown below. This example uses the ISPF LMCOPY command, from a REXX exec, to copy the listing from one data set to another, specifying the PACK option.

```
//PACKLIST EXEC PGM=IKJEFT01
//SYSPROC DD DSN=rexxy-exec-library,DISP=SHR
//ISPPLIB DD DISP=SHR,DSN=ISP.SISPENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSSLIB
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPMLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPLOG DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//ISPLIST DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SYSTSPTT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SYSPRINT DD SYSOUT=*
//LMIN DD DISP=SHR,DSN=compile-step-listing-data-set
//LMOUT DD DISP=SHR,DSN=packed-listing-data-set
//ISPPROF DD SPACE=(TRK,(5,1,4)),UNIT=SYSALLDA,
//          DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=6160)
//SYSTSIN DD *
ISPSTART CMD(LMCOPY member-name)
/*
```

where both *compile-step-listing-data-set* and *packed-listing-data-set* are names of PDS(E) data sets, without member specification, and *member-name* is the member name of the compiler listing (usually matching the program name being compiled).

## ISPF-packed compiler listings

The LMCOPY REXX exec, which must reside in the *rexxy-exec-library* data set as member LMCOPY:

```
/* LMCOPY REXX */
Address ISPEEXEC
Arg member
'LMINIT DATAID(INDD) DDNAME(LMIN) ENQ(SHRW)'
'LMINIT DATAID(OUTDD) DDNAME(LMOUT) ENQ(SHRW)'

'LMCOPY FROMID('INDD') TODATAID('OUTDD')
    FROMMEM('member') TOMEM('member') REPLACE PACK'

'LMFREE DATAID('INDD')
'LMFREE DATAID('OUTDD')
```

Although ISPF PACK saves DASD space used by compiler listings, much more space (approximately half the requirement) can be saved by converting the compiler listing into a LANGX side file (see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*, chapter “IPVLANGX compiler listing to side file conversion utility”, section “Creating side files using IPVLANGX” for information on how to do this), and then discarding the compiler listing. If you need to, then you can later reprint most of the original compiler listing information from the LANGX side file using the IPVLANGP utility (see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* for details).

---

## Chapter 21. Customizing the CICS environment

There are three exit points at which Fault Analyzer can be invoked for transaction abends under CICS. These are as follows:

### XPCABND

Global user exit using program IDIXCX52 or IDIXCX53. This exit is the main exit provided to invoke Fault Analyzer for CICS transaction fault analysis.

### XDUREQ

Global user exit using program IDIXCX52 or IDIXCX53. This exit can be used to invoke Fault Analyzer for CICS dumps generated from an EXEC CICS DUMP command.

The analysis performed by Fault Analyzer at this exit point is the same as for the XPCABND exit point.

### LE Exit

LE abnormal termination exit using program IDIXCCEE. This exit is only effective with Language Environment based application programs when the CEECXTAN exit has been set.

CICS AKCS abends can be analyzed using this exit, if both of the following are true:

- The failing program is LE enabled
- An entry exists in the CICS dump table for AKCS, specifying that a transaction dump is to be taken.

The first two of these exits are CICS global user exit points, and Fault Analyzer is enabled and disabled at these points using CICS calls. By default, these exit points are not enabled in a CICS region. They are enabled either by adding an entry to the CICS PLT (see “Adding the required programs to the startup PLT” on page 303), or by using the supplied CFA transaction once CICS has initialized (see “Controlling CICS transaction abend analysis” on page 305).

The LE abnormal termination exit, however, requires a modification to LE (see “Configuring Language Environment for CICS to invoke Fault Analyzer” on page 302), and hence its effect is system wide. Fault Analyzer provides a mechanism for controlling the use of this exit at a CICS region level, but in order for this mechanism to work, the LE exit must first be enabled system wide. Once enabled at a system wide level, then the initial setting in a CICS region is enabled.

IDI.SIDIAUTH needs to be added to the DFHRPL concatenation of the CICS JCL for any of the above exits to be successfully enabled.

To use Fault Analyzer with CICS, you need to perform the following steps:

— 1. **Configure Language Environment for CICS to invoke Fault Analyzer**

For details, see “Configuring Language Environment for CICS to invoke Fault Analyzer” on page 302.

— 2. **Define the required programs to your CICS system**

For details, see “Defining required programs to CICS” on page 302.

— 3. **Add the required programs to your startup PLT**

## Customizing the CICS environment

For details, see “Adding the required programs to the startup PLT” on page 303.

### — 4. Add the required programs to your shutdown PLT

For details, see “Adding the required programs to the shutdown PLT” on page 303.

### — 5. Define a transaction for Fault Analyzer

For details, see “Enabling dynamic control of analysis of CICS transaction abends” on page 303.

---

## Configuring Language Environment for CICS to invoke Fault Analyzer

Fault Analyzer provides a Language Environment abnormal termination exit for CICS, IDIXCCEE, as another method of invoking Fault Analyzer to the CICS XPCABND global exit—for more information, see “CICS LE abnormal termination CEECXTAN CSECT exit IDIXCCEE” on page 227. To enable this exit, you must add it to the CEECXTAN CSECT for Language Environment for CICS. To do this, make a copy of the CEEWCEXT softcopy sample member in the CEE.SCEESAMP data set. Make the changes suggested in the sample member and replace the lines:

```
<<< REPLACE THESE 2 LINES WITH A COPY OF CEECXTAN  
AND OVERRIDE AS DESIRED >>>
```

with

```
CEEEXAHD      ,User exit header  
CEEEXART    TERMxit=IDIXCCEE  
CEEEXAST      ,Terminate the list
```

For general information about implementing an Language Environment abnormal termination exit for CICS, refer to the *Language Environment Customization* book.

**Note:** The non-CICS version of this exit, CEEEXTAN, installed with sample job CEEWDEXT in data set CEE.SCEESAMP, is very similar to this exit. If you are going to install both, make sure that you double-check that you have used the correct names and not installed the same exit twice.

---

## Defining required programs to CICS

The following programs and BMS map must be defined to your CICS system, unless CICS program auto install is active:

- IDIPLT
- IDIPLTD
- IDIPLTS
- IDIXCX52
- IDIXCX53
- IDIXFA
- IDIXMAP (BMS map)

These programs are all assembler programs, and should be defined in a group that is included in a group list used during CICS startup.

The sample job shown in “Sample definition job” on page 304 includes definitions for all of the above programs.

---

## **Adding the required programs to the startup PLT**

To have Fault Analyzer install at CICS startup, add the program name, IDIPLT, to your startup PLT. IDIPLT enables either IDIXCX52 or IDIXCX53, depending on your version and release of CICS, as an XPCABND global user exit during CICS startup.

**Note:** For CICS open TCB users, adding the Fault Analyzer IDIPLT program name to the startup PLT is mandatory.

IDIPLT should be placed at the end of the PLT list, so that Fault Analyzer is not invoked before normal CICS services are available, should an abend occur in another PLT program. You can also enable Fault Analyzer at the XDUREQ global user exit by adding program name IDIPLTD to your startup PLT.

IDIPLTS can be included in the startup PLT if you require SDUMP screening to be installed.

---

## **Adding the required programs to the shutdown PLT**

To ensure that all Fault Analyzer activity is correctly terminated, it is necessary to add the following entry to your CICS shutdown PLT:

`DFHPLT TYPE=ENTRY,PROGRAM=IDIPLT`

The entry should be added before the DFHDELIM entry (shutdown phase 1).

As well as ensuring correct Fault Analyzer termination, IDIPLT also disables Fault Analyzer in all the currently enabled CICS exit points. This disablement prevents any subsequent abend analysis occurring during CICS shutdown. If analysis is required during shutdown, then do not add IDIPLT to your shutdown PLT. Note that it is then possible for system 33E abends to occur during shutdown.

---

## **Enabling dynamic control of analysis of CICS transaction abends**

When Fault Analyzer is installed, there is the option to also install a control transaction (called CFA in this document, however, you can name it as you desire) that lets you control dynamically the behavior of Fault Analyzer under CICS. To install the CFA transaction, define a CICS transaction for program IDIXFA in the same group as used for the above program definitions. You can use the CEDA transaction to do this, with default values for the parameters.

The CFA transaction should be prioritized high enough to enable it to be used to disable Fault Analyzer quickly in case of unexpected problems.

---

## **Using CFA to FORCEPURGE the currently analyzed task**

One of the features provided by program IDIXFA (transaction CFA) is the ability to FORCEPURGE the task that is currently being analyzed by Fault Analyzer.

If the CICS region in which IDIXFA executes is running with transaction isolation ACTIVE, then program IDIXFA must be defined with EXECKEY(CICS) in order for this feature to be available.

## SVC dump screening

### SVC dump screening

A feature is provided with the CFA transaction that can be used to improve performance when capturing SVC dumps of CICS regions. The feature, which can be installed or uninstalled using the CFA transaction, provides screening of the SDUMP SVC (SVC 51) to ensure that the SDATA dump options XESDATA and GRSQ are turned off while the dump is being taken.

**Note:** If you have other tools or products which provide screening of the SDUMP SVC (SVC 51), then it is recommended not to install the Fault Analyzer dump screening - either via the CFA transaction or by the IDIPLTS CICS PLT program. If another screening program is found to be installed when attempting to install Fault Analyzer dump screening, then the Fault Analyzer code is not installed and a message is issued.

### Sample definition job

Figure 139 shows a sample batch job that can be used to define all of the above mentioned programs and transaction to CICS. Replace data set names shown with *xxx* prefix with the correct names for your installation and *list-name* with the appropriate CICS startup SIT GRPLIST name. The group name FA has been chosen for the sake of this example, but can be changed if you desire.

```
//IDICICS JOB ...
//IDICICS EXEC PGM=DFHCSDUP,REGION=1024K,
//           PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB  DD DISP=SHR,DSN=xxx.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=xxx.DFHCSD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
      DEF PROGRAM(IDIPLT) GROUP(FA) EXECKEY(CICS)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIPLTD) GROUP(FA) EXECKEY(CICS)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIPLTS) GROUP(FA) EXECKEY(CICS)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIXCX52) GROUP(FA)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIXCX53) GROUP(FA)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIXCCEE) GROUP(FA)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIXFA) GROUP(FA) EXECKEY(CICS)
        CEDF(NO) DATALOCATION(ANY)
        CONCURRENCY(QUASIRENT)
      DEF PROGRAM(IDIVPCLE) GROUP(FA)
        CONCURRENCY(QUASIRENT)
      DEF TRANSACTION(CFA) GROUP(FA)
        PROGRAM(IDIXFA) TASKDATALOC(ANY)
        SHUTDOWN(ENABLED)
      DEF MAPSET(IDIXMAP) GROUP(FA)
        ADD G(FA) L(list-name)
/*
/*
```

Figure 139. Sample Fault Analyzer CICS program and transaction definition job

The above sample job is provided as member IDISCICS in data set IDI.SIDISAM1.

In order for Fault Analyzer to be invoked under CICS, it is necessary to add IDI.SIDIAUTH to the DFHRPL concatenation.

CICS tracing must be active for Fault Analyzer to display CICS trace information.

If CICS is used without LE in the LINKLIST, it is necessary to install the IDILEDS USERMOD as described in "Identifying the LE runtime library" on page 257.

The ABCODE keyword must be used on an EXEC CICS ABEND statement in order for Fault Analyzer to be invoked. For example:

```
EXEC CICS ABEND ABCODE('abcd') END-EXEC
```

If the NODUMP keyword is used on an EXEC CICS ABEND statement, then Fault Analyzer only performs analysis if invoked via the IDIXCCEE exit.

---

## Controlling CICS transaction abend analysis

Once the CFA transaction is installed (you might have chosen to install it under a different name, as per the above), then it can be used to install or uninstall the following Fault Analyzer invocation exits:

- XPCABND CICS global user exit
- XDUREQ CICS global user exit
- LE abnormal termination exit
- XEIIN global user exit

In addition, the CFA transaction can be used to install or uninstall the Fault Analyzer SDUMP screening feature.

Prior to installing either the XPCABND or XDUREQ exits, the CFA transaction issues a CICS NEWCOPY command for program IDIXCX52 and IDIXCX53 if both exits are in the "Uninstalled" state. Hence, to load a new copy of IDIXCX52 or IDIXCX53, for example after applying maintenance, use the CFA transaction to uninstall the XPCABND and XDUREQ exits, and then reinstall either or both exits as required.

There are two ways to interact with the CFA transaction; either from a CICS terminal, or from the MVS console. Each of these are described in the following sections.

### Using CFA from a CICS terminal

To use the CFA transaction from a CICS terminal, simply enter CFA. You can optionally pass a command parameter as described in "Using CFA from an MVS console" on page 308. The initial display is similar to the following:

## Controlling CICS transaction abend analysis

```
Fault Analyzer Control Transaction

Options: I=Install U=Uninstall
          Current Status/Error Message
- XPCABND      Installed
- XDUREQ       Installed
- LE Exit       Installed
- SDUMP Screening Installed
- XEIIN        Uninstalled

      Current    HWM   Setting   MWS
Active     0000    0000    0002
Waiting    0000    0000    0020    0123

IDITRACE OFF DeferredReport ON

PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

Figure 140. Sample CFA transaction display

Initially, the display shows the current status of the CICS Fault Analyzer exit points, plus details of any active and waiting Fault Analyzer tasks. By entering an I or U (for Install or Uninstall) next to a specific exit point, its status can be changed accordingly. If there is an active analysis task (as shown in the example above), then a CICS TASK FORCEPURGE can be issued for that task by entering an F in the input field next to the active task details. This function is only possible if CICS transaction isolation is INACTIVE, or if ACTIVE, that the IDIXFA program is defined to have an EXECKEY of CICS.

PF9 can be used to display a list of IVP tests. For details, see “Verifying the customization of Fault Analyzer under CICS” on page 329.

The PF11 function is described in “IDITRACE under CICS” on page 307.

For help information about a specific CICS exit, press PF1 on the main panel with the cursor on an exit selection field.

If pressing PF4, then the Fault Analyzer Exit Options display is shown. This display provides information about the current default options that are in effect for the CICS region. An example of this display is shown in Figure 141 on page 307.

Fault Analyzer Exit Options		
Description	Option	Setting
Debug trace . . . . .	ZZDEBUG . . . . .	OFF
Suppress Msg: IDI0034I . . . . .	QUIET(IDI0034I) . . .	OFF
IDI0066I . . . . .	QUIET(IDI0066I) . . .	OFF
IDI0118W . . . . .	QUIET(IDI0118W) . . .	OFF
Transaction Dump Table Check . . . . .	CICSDUMPTABLEEXCLUDE	ON
Retain CICS dumps . . . . .	RETAINCICSDUMP . . .	ALL
Fast duplicate minutes . . . . .	NODUP . . . . .	00000
Include EXEC CICS DUMP . . . . .	IECD . . . . .	OFF

PF3=Exit

Figure 141. Sample CFA Exit Options display

### Clearing the NoDup(CICSFAST(...)) recording area

Under CICS, Fault Analyzer maintains a table of recent abends which it has processed, called the FND (fast-no-dup) area. This area is used in conjunction with the NoDup(CICSFAST(...)) option. PF5 can be used to clear this area, which means that no previous abends are used in subsequent FAST duplicate determination. You are prompted to press PF5 again to confirm that you want to clear the FND area.

### IDITRACE under CICS

PF11 from the CFA transaction can be used to turn the Fault Analyzer debugging trace, IDITRACE, ON or OFF. The trace is written to the IDITRACE DDname, dynamically allocated to the JES spool. For example, entering '?' against a CICS region under SDSF, might result in the following display with the IDITRACE output selectable with 'S' at **1**:

## Controlling CICS transaction abend analysis

SDSF JOB DATA SET DISPLAY - JOB AS650F1 (JOB31639)							DISPLAY RESET	SCROLL ==> CSR
COMMAND INPUT ==>			PREFIX=*= DEST=(ALL) OWNER=SIMCOCK SYSNAME=					
NP	DDNAME	StepName	ProcStep	DSID	Owner	C Dest	Rec-Cnt	Page
	JESMSGLG	JES2		2	SIMCOCK	X	4	
	JESJCL	JES2		3	SIMCOCK	X	259	
	JESYMSG	JES2		4	SIMCOCK	X	2,173	
	SYSPRINT	AS650F1	AS650F1	103	SIMCOCK	X	126	
	MSGUSR	AS650F1	AS650F1	105	SIMCOCK	X	2,328	
	PLIMSG	AS650F1	AS650F1	106	SIMCOCK	X	0	
	DFHCXRF	AS650F1	AS650F1	107	SIMCOCK	X	641	
	COUT	AS650F1	AS650F1	108	SIMCOCK	X	0	
	CEEMSG	AS650F1	AS650F1	109	SIMCOCK	X	0	
	CEEOUT	AS650F1	AS650F1	110	SIMCOCK	X	0	
	PRINTER	AS650F1	AS650F1	116	SIMCOCK	X	0	
1	IDITRACE	AS650F1	AS650F1	117	SIMCOCK	X	172	
	CRPO	AS650F1	AS650F1	119	SIMCOCK	X	0	

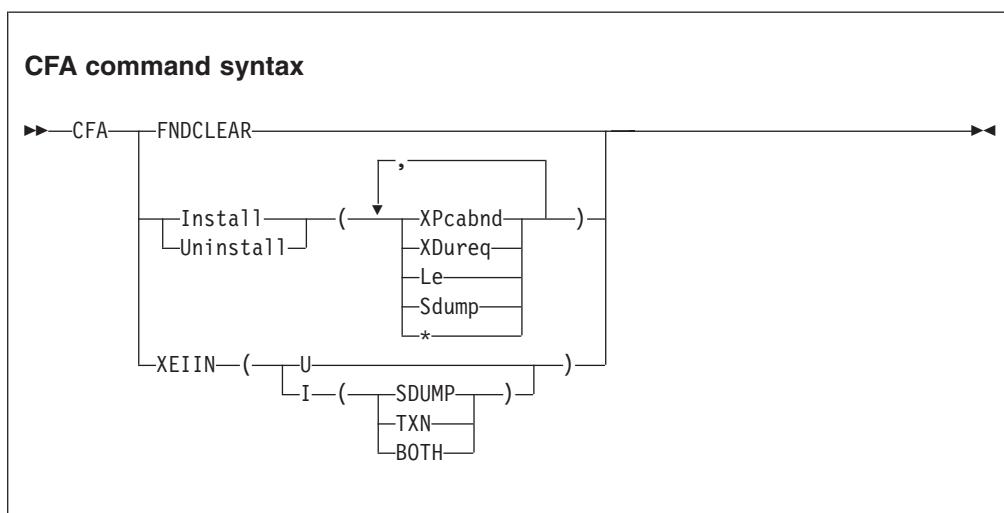
Figure 142. Sample CICS region SDSF Job Data Set display

**Note:** The CFA transaction can only successfully turn off IDITRACE if there is not an IDITRACE DDname in the CICS JCL. This approach is in accordance with the general principle adhered to by Fault Analyzer, which is to allow JCL-specified DDnames to take precedence over dynamic allocations or unallocations.

If the CICS JCL contains an IDITRACE DD statement, then the trace is turned on at the time of starting CICS, and it remains turned on until CICS is stopped, regardless of any attempt to turn the trace on or off with the CFA transaction.

## Using CFA from an MVS console

To use the CFA transaction from an MVS console, issue the MODIFY (F) command with the CFA command parameter. The CFA command syntax is shown below.



where:

### **FNDCLEAR**

Clears the FND area (see description in “Using CFA from a CICS terminal” on page 305).

### **INSTALL | UNINSTALL**

Installs or uninstalls specific Fault Analyzer CICS exits (other than the XEIIN global user exit). Each of the options below can be specified in any sequence, enclosed in parenthesis:

#### **XPCABND**

Installs or uninstalls the CICS XPCABND global user exit.

#### **XDUREQ**

Installs or uninstalls the CICS XDUREQ global user exit.

**LE** Installs or uninstalls the CICS IDIXCCEE LE abnormal termination exit.

#### **SDUMP**

Installs or uninstalls CICS SVC dump screening.

**\*** Installs or uninstalls all of the above (XPCABND, XDUREQ, LE and SDUMP).

**XEIIN** Installs or uninstalls the Fault Analyzer CICS XEIIN global user exit.

**U** Uninstalls the XEIIN exit.

**I** Installs the XEIIN exit for the specified operation:

#### **SDUMP**

Installs the CICS XEIIN global user exit for system dumps.

**TXN** Installs the CICS XEIIN global user exit for transaction dumps.

**BOTH** Installs the CICS XEIIN global user exit for both system and transaction dumps.

### **Examples:**

- To uninstall all exits from CICS region CICS01, enter the command:  
`F CICS01,CFA U(*)`
- To install the LE exit for use by CICS region TESTCICS, enter the command:  
`F TESTCICS,CFA I(L)`
- To install the XPCABND and XDUREQ exits for use by CICS region PRODCICS, enter the command:  
`F PRODCICS,CFA I(XP,XD)`
- To clear the FND area in CICS region MYCICS, enter the command:  
`/F MYCICS,CFA FNDCLEAR`
- To uninstall the XEIIN exit from CICS region CICSP1, enter the command:  
`/F CICSP1,CFA XEIIN(U)`
- To install the XEIIN exit for system dumps by CICS region CICSA, enter the command:  
`/F CICSA,CFA XEIIN(I(SDUMP))`

## **Ensuring transaction abend analysis is not suppressed by DUMP(NO)**

If the active transaction definition for an abending transaction has the DUMP(NO) option specified, then CICS does not call the XPCABND global user exit, and hence Fault Analyzer is not invoked. To check the DUMP setting for a transaction, do one of the following:

- Use the CEDA transaction to view the transaction definition in question and check the DUMP(YES | NO) setting. Care should be taken when using this method as there might be multiple definitions of the same transaction, and hence the order in which the definitions are installed by CICS is important.
- Check the active transaction definition in a dump.
- Use the CICS supplied transaction, CECI, to check the DUMP setting for the active transaction. This checking can be done by issuing the following command:  
`CECI INQUIRE TRANSACTION(nnnn)`

where *nnnn* is the transaction ID in question.

Having issued this command, the displayed DUMPING value has the following meaning:

- A value of 00186 means DUMP(YES).
- A value of 00187 means DUMP(NO).

---

## **CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)**

In certain circumstances where Fault Analyzer has detected an abend as a CICS NoDup(CICSFAST) duplicate (see “NoDup” on page 482 for information about the NoDup option), it might be desirable to override this detection, and hence instigate normal abend analysis. These circumstances might, for example, be for specific abends or transactions, or combinations of these, for which a full analysis is required. To accommodate this situation, there exists a NoDup(CICSFAST) assembler exit, IDINDFUE. Fault Analyzer attempts to load this exit program prior to issuing the CICS NoDup(CICSFAST) message, IDI0066I. If the load is successful, then program IDINDFUE will be invoked.

### **Invocation**

IDINDFUE must be written in assembler language. It is called by using the OS linkage convention.

IDINDFUE might be called from IDIXCCEE or IDIXCX53, that is, from the CICS XPCABND Global User Exit (GLUE). It must adhere to the CICS rules for coding an XPCABND GLUE program. See the CICS customization guide for details. When you are assembling IDINDFUE, do not use the CICS command-level translator.

The exit program must be:

- Link-edited into a load module named IDINDFUE.
- Placed into the DFHRPL concatenation of your CICS region if auto load is active.
- Included in the PPT if auto load is not active.

### **Entry specifications**

Contents of register on entry to IDINDFUE:

#### **Register**

#### **Contents**

- 1 Address of input parameter list (see below).

## CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)

- 13 Address of 72-byte register save area.
- 14 Return address.

**Input parameter list:** R1 must contain the address of a parameter list, consisting of a list of addresses (OS linkage). Each address in the parameter list points to a parameter as described.

Table 9. IDINDFUE input parameters

Parameter	Number of bytes	Description
Parameter 1	256	User work area
Parameter 2	4	Abend code
Parameter 3	4	Transaction ID
Parameter 4	8	Abending program name

## Return specifications

Contents of register on return from IDINDFUE:

### Register

#### Contents

- 0-14 Unchanged.
- 15 Return code:
  - 0 Indicates that the abend should continue to be treated as a duplicate.
  - 1 Indicates that abend analysis should proceed, that is, overriding the NoDup(CICSFAST) detection.

**Note:** If an IDINDFUE exit passes back a return code of 1, indicating that analysis should be performed, no NoDup(NORMAL) duplicate detection is performed.

## Example

A softcopy of the sample user exit shown in the following is also provided as member IDINDFUE in data set IDI.SIDISAM1. This sample assembler exit sets R15 to 1 if the transaction ID is FTN1, and the abend code is FABN.

```
IDINDFUE CSECT
    PRINT ON,NOGEN
    STM   R14,R12,12(R13)
    LR    R12,R15
    USING IDINDFUE,R12
    L     R4,0(,R1)           Work Area
    L     R2,4(,R1)           Abend code
    L     R3,8(,R1)           Transaction ID

    SR   R15,R15             Assume no continuance.
    CLC  0(4,R2),=C'FABN'   Check Abend code
    BNE  RETURN
    CLC  0(4,R3),=C'FTN1'   Check Transaction ID
    BNE  RETURN
    LA   R15,1                Indicate analysis to be performed
*
RETURN DS  0H
        L  R14,12(,R13)
        LM R0,R12,20(R13)
        BR R14
        END IDINDFUE           Return to Fault Analyzer
```

### CICS trace considerations

When Fault Analyzer gathers the trace entries for the abending task, a storage area is used to copy the trace entries into. The size of this storage area, and hence the number of trace entries copied, is determined as follows:

- For CICS TS 5.1 and above, the CICS system initialization parameter, TRTRANSZ is used.
- For earlier CICS releases, minimum of 64K is used, unless the CICS system initialization parameter, TRTRANSZ, specifies a value higher than 64K, in which case that value is used, up to a maximum of 256K.

### Preventing LE from causing the CICS trace to wrap

When a CICS transaction abends and Language Environment is active in the abending enclave, Language Environment by default writes diagnostic information to a transient data queue named CESE. This situation occurs if the IBM-supplied Language Environment default runtime option TERMTHDACT(TRACE) is in effect. Because these diagnostics are recorded before Fault Analyzer receives control to process the abend, the CICS trace table is liable to wrap around, and application trace data might therefore be lost. Depending on your level of MVS, it is recommended that the TERMTHDACT option is set to one of the following:

#### TERMTHDACT(TRACE,CICSDDS,...)

This value causes Language Environment to write its diagnostics to the CICS transaction dump data set.

#### TERMTHDACT(QUIET)

This value suppresses most of the Language Environment diagnostics.

### Specifying CICS options through the IDIOPTS DDname

To avoid the need to recycle CICS in case compiler listing or side file data sets change, specify these via the DataSets option in a user options file that is pointed to by the IDIOPTS DDname. For details, refer to “DataSets” on page 457 and “User options file IDIOPTS” on page 454.

It is also better to use the IDIOPTS DDname pointing to a data set and containing  
DataSets(IDIHIST(*history-file-dsn*))

than using

//IDIHIST DD DISP=SHR,DSN=*history-file-dsn*

in CICS JCL.

If you use

//IDIHIST DD DISP=SHR,DSN=*history-file-dsn*

then JCL allocation holds an ENQ on the history file data set name for the full time when CICS is running. However, if you use

DataSets(IDIHIST(*history-file-dsn*))

then the allocation is dynamically obtained and released when required by Fault Analyzer. As a result, history file maintenance and renames can be done while a CICS system that has written to it is still running. The

DataSets(IDIHIST(*history-file-dsn*))

option can even be changed to point to a new history file while CICS is running and takes effect when the next fault entry is created.

The IDIOPTS data set used must be a PDS(E) in order to permit update using ISPF EDIT while CICS is running.

---

## Language Environment abend considerations

If an abend occurs in Language Environment while trying to recover from a transaction abend under CICS, then Fault Analyzer is not invoked. This lack of invocation is because CICS normal behavior for these types of abends is to not drive the XPCABND exit. However, by enabling the CICS LE abnormal termination CEECXTAN CSECT exit IDIXCCEE, which is provided with Fault Analyzer, these types of abends are still analyzed. See “Configuring Language Environment for CICS to invoke Fault Analyzer” on page 302 for details on the enablement of this exit.

---

## Capture of abends running on CICS user key open TCBs (L9 TCBs)

In order for Fault Analyzer to perform abend analysis of a CICS transaction running on an open (L9) TCB, it is necessary to have the Fault Analyzer XPCABND user exit enabled—see “Controlling CICS transaction abend analysis” on page 305 for details about enabling this exit.

---

## Installing the MVS post-dump exit IDIXTSEL

The Fault Analyzer post-dump exit, IDIXTSEL, is installed in the IEAVTSEL installation exit list. The exit, which is only invoked for SVC dumps, is installed by the USERMOD, IDIWTSEL. It is required to register CICS system abend dumps, recovery fault recording SDUMPs and to facilitate the capturing of Java faults.

**Note:** SVC dumps taken as the result of SLIP traps with ACTION=SVCDUMP are not captured by this exit.

To install this USERMOD, edit and submit the sample job IDIWTSEL. This sample job includes IDIXTSEL in the IEAVTSEL installation exit list. If you have other exits defined in this list, add the IDIXTSEL exit last.

For more information about adding exits to IEAVTSEL, see *MVS Installation Exits*.

To activate this change, IPL again or cancel the DUMPSRV address space so that it restarts with the new exit.

For dump registration via this exit to occur, it is necessary to also start the Fault Analyzer IDIS subsystem—for information on this, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239.

---

## Storage requirements

Insufficient storage for Fault Analyzer under CICS can result in S878 abends, which might bring down the CICS region.

For information about storage requirements under CICS, see “Storage recommendations” on page 222

### Maximizing CICS transaction abend analysis performance

To maximize CICS transaction abend analysis performance, the following should be considered:

- Using the Fault Analyzer IDIS subsystem to manage the access to history file \$\$INDEX members. For details, see “Caching of history file \$\$INDEX data” on page 240.
- Using the DeferredReport option. For details, see “DeferredReport” on page 463. You can specify this option via the IDIOPTS DDname, as explained in “Specifying CICS options through the IDIOPTS DDname” on page 312, if you want to make it applicable to CICS only.

### Implementing an XEIIIN global user exit

The XEIIIN global user exit can be useful if an overlay is not being detected until CICS task termination is doing the final freeing of storage. CICS can detect a Storage Accounting Area (SAA) overlay either during an explicit FREEMAIN or during task termination. If detected at task termination, the task is not abended by CICS and hence Fault Analyzer is not invoked. CICS attempts to take a system dump, with dump code SM0102, however if CICS system dumps are being suppressed, further analysis is not possible.

To assist with SAA overlays Fault Analyzer provides a program that can optionally be installed in the XEIIIN global user exit. If installed, this program, at each EXEC CICS RETURN statement, validates all the CICS storage accounting areas for the current task. If an overlay is found, then one of two actions can be performed:

1. A call can be made to take a CICS transaction dump with dumpcode IDIS.  
If Fault Analyzer is installed in the XDUREQ global user exit, then this transaction dump is analyzed in the normal manner with a history file fault entry being created.
2. A SDUMP of the CICS region can be taken.  
The request to take this SDUMP is made by the IDIS subsystem, and as such, it is not affected by the CICS system dump suppression setting. If a SDUMP is captured, then it can then be analyzed interactively by way of the **File** menu option 5.

Enabling the Fault Analyzer XEIIIN global user exit program, and setting which action should be performed, is done using the Fault Analyzer supplied CFA transaction. Unlike the other Fault Analyzer CICS exits, there is no option to enable the XEIIIN exit during CICS PLT processing.

When invoked, the initial CFA display is similar to the following:

## Implementing an XEIN global user exit

```
Fault Analyzer Control Transaction
Options: I=Install U=Uninstall
          Current Status/Error Message
- XPCABND      Installed
- XDUREQ       Installed
- LE Exit       Installed
- SDUMP Screening Installed
- XEIN         Uninstalled

          Current   HWM   Setting   MWS
Active     0000    0001    0001
Waiting    0000    0000    0017    0000

IDITRACE OFF DeferredReport ON
PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

From this display, an "I" can be entered next to XEIN to install the exit. A screen similar to the following is then displayed:

```
Fault Analyzer Control Transaction
Options: I=Install U=Uninstall
          Current Status/Error Message
- XPCABND      Installed
- XDUREQ       Installed
- LE Exit       Installed
- SDUMP Screening Installed
- XEIN         Installed   N TXN dump N SDUMP Set at least one option to Y

          Current   HWM   Setting   MWS
Active     0000    0001    0001
Waiting    0000    0000    0017    0000

IDITRACE OFF DeferredReport ON
PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

From here, a "Y" can be entered next to TXN dump or SDUMP to set the appropriate action that should be taken if a SAA overlay be detected.

The I (Install) option must be used each time the TXN and SDUMP fields are changed.

**Note:** Although there is no overhead in having any other Fault Analyzer exit installed/enabled, the XEIN exit is invoked by CICS on every EXEC CICS command, and hence does incur a small overhead. As such, it is recommended to only use the XEIN exit when needing to investigate SAA overlays.

## **Implementing an XEIN global user exit**

---

## Chapter 22. Customizing the DB2 environment

This section contains information specific to Fault Analyzer in DB2 environments.

---

### Binding DB2

If Fault Analyzer is to be run against abends occurring in applications that use DB2, then you must ensure that the DB2 Call Level Interface (CLI) is installed and the required setup has been performed to bind plan DSNACLI. The DSNACLI plan can be created using the sample job in member DSNTIJCL of the DB2 SDSNSAMP data set. Refer to the *DB2 for OS/390 Call Level Interface Guide and Reference* for further information.

Ensure that the Fault Analyzer IDIS subsystem subsystem has SELECT authority to the required DB2 system catalog tables—see “IDIS subsystem requirements for DB2” on page 243 for details.

---

### DB2 and Language Environment

If the CEEEXTAN LE abnormal termination exit (IDIXCEE) has not been installed, then either TERMTHDACT(UATRACE), TERMTHDACT(UADUMP), TERMTHDACT(UAONLY), or TERMTHDACT(UAIMM) must be passed through to LE in order to have Fault Analyzer invoked for the DB2 abend.

Below is a COBOL/DB2 example that illustrates how LE options can be passed:

```
//MYJOB    JOB
//STEP1    EXEC PGM=IKJEFT01
//DBRMLIB  DD DSN=TEST.DB2.DBRLIB.DATA,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//IDIREPRT DD SYSOUT=*
//IDIHIST   DD DISP=SHR,DSN=TEST.DB2.HIST
//IDILCOB   DD DISP=SHR,DSN=TEST.LISTING.DB2.COBOL
//SYSIN    DD *
//SYSTSIN  DD *
DSN SYSTEM(DSN1)
BIND PLAN(DSNTEST1) QUALIFIER(DSN8510) MEMBER(DACBD001) -
ACT(REP) ISOLATION(CS)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA51) -
LIB('DSN510.RUNLIB.LOAD')
RUN PROGRAM(DACBD001) PLAN(DSNTEST1) -
LIB('CTEST.DB2.LOAD') -
PARMS('/TERMTHDACT(UADUMP) ')
END
/*
```

---

### Improving Fault Analyzer DB2 performance

The following information is applicable to DB2 versions prior to DB2 V10. For DB2 V10 or later, this information can be ignored.

DB2 does not have an index defined on the SYSIBM.SYSDBRM catalog table. Fault Analyzer accesses the SYSIBM.SYSDBRM catalog table whenever it performs analysis for a DB2 fault. To avoid the possibility of poor performance when

## Improving Fault Analyzer DB2 performance

accessing SYSIBM.SYSDBRM, you can create a user-defined index on the SYSIBM.SYSDBRM catalog table. The non-unique index should specify the following columns (in order):

PLNAME  
NAME

The following sample DDL can be used to define the index:

```
CREATE TYPE 2 INDEX nnnnnnn.DBRMX           1
  ON SYSIBM.SYSDBRM
  (
    PLNAME ASC,
    NAME ASC
  )
  USING STOGROUP mmmmmmmm                   1
    PRIQTY pp
    SECQTY ss                                2
  )
CLOSE NO;
```

Notes:

- 1** Change the name of the index (*nnnnnnn*) and storage group (*mmmmmmmm*) to suit your requirements. For example, use index name SYSIBM and storage group STOGROUP.
- 2** Change the primary (*pp*) and secondary (*ss*) quantities to suit your requirements. For example, use 250 for both primary and secondary quantity.

A sample job, IDISDB2X, is distributed in IDI.SIDISAM1 to help you do this.

---

## Chapter 23. Customizing the IMS environment

This section contains information specific to Fault Analyzer in IMS environments.

---

### IMS and Language Environment

If the CEEEXTAN LE abnormal termination exit (IDIXCEE) has not been installed, then either TERMTHDACT(UATRACE), TERMTHDACT(UADUMP), TERMTHDACT(UAONLY), or TERMTHDACT(UAIMM) must be passed through to LE in order to have Fault Analyzer invoked for the IMS abend.

Below is a COBOL/IMS example that illustrates how LE options can be passed by linking a CEEUOPT CSECT into the load module being executed:

```
//IMSLE1 JOB ...
/*
/*      STEP 1: ASSEMBLE CEEUOPT CSECT
/*
//HLASM    EXEC PGM=ASMA90,PARM='LINECOUNT(0)'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=(,PASS),UNIT=SYSALDDA,SPACE=(TRK,(1,5))
//SYSLIN    DD DISP=(,PASS),UNIT=SYSALDDA,SPACE=(TRK,(1,5,1)),DSN=&TEMP(CEEUOPT)
//SYSLIB    DD DSN=CEE.SCEEMAC,DISP=SHR
//          DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN     DD *
          TITLE 'CEEUOPT'
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
          CEEXOPT TERMTHDACT=(UADUMP)
          END
/*
/*      STEP 2: COMPILE COBOL PROGRAM
/*
//COBCOMP  EXEC IMSCOBOL
//COB.SYSIN  DD DSN=DA.IMSSAMP.COBOL(BATCHJ2),DISP=SHR
//COB.SYSPRINT DD DSN=DA.LISTING.COBOL(BATCHJ2),DISP=SHR
//LKED.FRED  DD DSN=*.HLASM.SYSLIN,DISP=OLD
//LKED.SYSIN  DD *
          Include FRED(CEEUOPT)
          NAME    BATCHJ2(R)
/*
/*
/*      STEP 3: RUN THE PROGRAM
/*
//PROGRUN  EXEC PROC=DLIBATCH,MBR=BATCHJ2,PSB=PSB1,COND=(4,LT),
//          DBRC=Y,MON=Y,FMT0=D,TIME=5
//          UNIT=3390,
//          DCB=BLKSIZE=6144
//SYSPRINT DD SYSOUT=*
//DFSIVD1  DD DISP=SHR,DSN=IMS.DFSIVD1
//DFSIVDII DD DISP=SHR,DSN=IMS.DFSIVDII
//DFSCTL   DD DISP=SHR,
//          DSN=IMS.PROCLIB(DFSSBPRM)
//IDIREPRT DD SYSOUT=*
//SYSTSIN  DD *
/*
```



---

## Chapter 24. Customizing the Fault Analyzer Japanese feature

This section is applicable to users of the Japanese feature of Fault Analyzer only.

---

### Allocating ISPF data sets

The following ISPF DDnames and data sets must be allocated in addition to those required for the base function of Fault Analyzer shown in Chapter 15, “Modifying your ISPF environment,” on page 247:

DDname	Data set name
<b>IDIIPJPN</b>	IDI.SIDIPJPN
<b>IDIIMJPN</b>	IDI.SIDIMJPN
<b>IDIIISJPN</b>	IDI.SIDISJPN
<b>IDIITJPN</b>	IDI.SIDITJPN

Typically, data sets for an ISPF application are allocated in either the TSO logon procedure, a program or an EXEC run prior to invoking ISPF, or dynamically (for example, in an EXEC) prior to invoking the application using the ISPF LIBDEF service.

When Fault Analyzer is invoked using the Language(JPN) option, then Fault Analyzer uses the ISPF LIBDEF service to logically place the data sets that are allocated to the IDIIPJPN, IDIIMJPN, IDIIISJPN, and IDIITJPN DDnames ahead of the data sets that are allocated to the ISPPLIB, ISPMLIB, ISPSLIB, and ISPTLIB DDnames. The stacking feature of the LIBDEF service is used to ensure that any data sets defined using LIBDEF prior to invoking Fault Analyzer are restored on exit.

If a LIBDEF for either ISPPLIB, ISPMLIB, ISPSLIB, or ISPTLIB is already active at the time of invoking Fault Analyzer, then the existing LIBDEF data sets are included in the new LIBDEF, after the IDIIPJPN, IDIIMJPN, IDIIISJPN, or IDIITJPN data sets. Because the maximum number of data sets that can be specified with LIBDEF when using the DATASET option is limited to 15, any data sets in excess of 14 that is already specified using LIBDEF, are not available. (This arithmetic assumes that only one data set is specified for the IDIIPJPN, IDIIMJPN, IDIIISJPN, and IDIITJPN DDname, as is normally the case.) Therefore, it is important that any Fault Analyzer base function ISPF data sets, that are specified using LIBDEF at the time of invoking Fault Analyzer are among those that are included in the Fault Analyzer established LIBDEF, otherwise ISPF failures might result due to untranslated members not being found.

---

### Setting the national language

The Language option (see “Language” on page 479) specifies the national language ID, which is used to select appropriate language-dependant data sets.

## **Setting the national language**

To make Japanese the default language for all users, specify the following option in your IDICNF00 parmlib member:

LANGUAGE(JPN)

---

## Chapter 25. Customizing the Fault Analyzer Korean feature

This section is applicable to users of the Korean feature of Fault Analyzer only.

---

### Allocating ISPF data sets

The following ISPF DDnames and data sets must be allocated in addition to those required for the base function of Fault Analyzer shown in Chapter 15, “Modifying your ISPF environment,” on page 247:

DDname	Data set name
<b>IDIIPKOR</b>	IDI.SIDIPKOR
<b>IDIIMKOR</b>	IDI.SIDIMKOR
<b>IDIISKOR</b>	IDI.SIDISKOR
<b>IDIITKOR</b>	IDI.SIDITKOR

Typically, data sets for an ISPF application are allocated in either the TSO logon procedure, a program or an EXEC run prior to invoking ISPF, or dynamically (for example, in an EXEC) prior to invoking the application using the ISPF LIBDEF service.

When Fault Analyzer is invoked using the Language(KOR) option, then Fault Analyzer uses the ISPF LIBDEF service to logically place the data sets that are allocated to the IDIIPKOR, IDIIMKOR, IDIISKOR, and IDIITKOR DDnames ahead of the data sets that are allocated to the ISPPLIB, ISPMLIB, ISPSLIB, and ISPTLIB DDnames. The stacking feature of the LIBDEF service is used to ensure that any data sets defined using LIBDEF prior to invoking Fault Analyzer are restored on exit.

If a LIBDEF for either ISPPLIB, ISPMLIB, ISPSLIB, or ISPTLIB is already active at the time of invoking Fault Analyzer, then the existing LIBDEF data sets are included in the new LIBDEF, after the IDIIPKOR, IDIIMKOR, IDIISKOR, or IDIITKOR data sets. Because the maximum number of data sets that can be specified with LIBDEF when using the DATASET option is limited to 15, any data sets in excess of 14 that is already specified using LIBDEF, are not available. (This arithmetic assumes that only one data set is specified for the IDIIPKOR, IDIIMKOR, IDIISKOR, and IDIITKOR DDname, as is normally the case.) Therefore, it is important that any Fault Analyzer base function ISPF data sets that are specified using LIBDEF at the time of invoking Fault Analyzer are among those that are included in the Fault Analyzer established LIBDEF, otherwise ISPF failures might result due to untranslated members not being found.

---

### Setting the national language

The Language option (see “Language” on page 479) specifies the national language ID, which is used to select appropriate language-dependant data sets.

## **Setting the national language**

To make Korean the default language for all users, specify the following option in your IDICNF00 parmlib member:

LANGUAGE(KOR)

---

## Chapter 26. Verifying the customization of Fault Analyzer

To verify the successful installation and customization of Fault Analyzer, instructions are provided in the following for different program languages and execution environments.

### Verifying the use of Fault Analyzer with assembler

To verify Fault Analyzer with assembler, edit and submit the sample job IDIVPASM in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job assembles and executes a program, which abends with a system abend code of 0C7.

Since this program is a non-LE program, Fault Analyzer is invoked via the MVS change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

**Japanese feature note**

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

**End of Japanese feature note**

**Korean feature note**

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

**End of Korean feature note**

A system abend 0C7 reason code X'0' occurred in module GO CSECT IDISASM1 at offset X'44'.

A program interruption code 0007 (Data Exception) is associated with this abend and indicates that:

A decimal digit or sign was invalid.

The cause of the failure was module GO CSECT IDISASM1. The Assembler source code that immediately preceded the failure was:

```
List  
Stmt #  
-----  
000037      CVB      R5,WorkD
```

A complete sample report from running this IVP is provided as member IDISRP03 in the IDI.SIDIDOC1 data set.

## Verifying the use of Fault Analyzer with assembler

An LE-compliant version of this IVP is provided as member IDIVPBLE in data set IDI.SIDISAM1—see “Verifying the IDIXCEE Language Environment exit enablement” on page 329 for more information.

## Verifying the use of Fault Analyzer with COBOL

This section is only applicable if you have COBOL installed at your site.

To verify Fault Analyzer with COBOL, edit and submit the sample job IDIVPCOB in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job compiles and executes a COBOL program, which abends with a return code of 3000.

As a result of the TER(UADUMP) LE option, and the provision of a SYSMDUMP DD statement, Fault Analyzer is invoked via the MVS change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

### End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

### End of Korean feature note

**Note:** Due to differences in version, release or maintenance level of the compiler used, program offset information might differ from the sample below.

A system abend 0C7 occurred in module IDISCBL1 program IDISCBL1 at offset X'41E'.

A program interruption code 0007 (Data Exception) is associated with this abend and indicates that:

A decimal digit or sign was invalid.

The cause of the failure was program IDISCBL1 in module IDISCBL1. The COBOL source code that immediately preceded the failure was:

```
Source  
Line #  
-----  
000029      CLEAR SECTION.  
000030      START001.  
000031      DIVIDE NUMBERX BY ERROR-COUNT GIVING BAD-RESULT.
```

The COBOL source code for data fields involved in the failure:

```
Source  
Line #  
-----  
000011      01  NUMBERX PIC 999999 COMP-3.  
000013          05  ERROR-COUNT PIC 999999 COMP-3.  
000016      01  BAD-RESULT PIC 99 COMP-3.
```

Data field values at time of abend:

```
BAD-RESULT    = X'0000'  
ERROR-COUNT   = X'C1C2C3C4' *** Cause of error ***  
NUMBERX       = 986888
```

A complete sample report from running this IVP is provided as member IDISRP01 in the IDI.SIDIDOC1 data set.

## Verifying the use of Fault Analyzer with PL/I

This section is only applicable if you have PL/I installed at your site.

Depending on your version of PL/I, there are two separate IVP jobs available:

- If using Enterprise PL/I, edit and submit the sample job IDIVPPL in data set IDI.SIDISAM1.
- If not using Enterprise PL/I, edit and submit the sample job IDIVPPLI in data set IDI.SIDISAM1.

Refer to the instructions in the sample jobs for more information.

Each job compiles and executes a PL/I program, which abends and terminates the job step with a return code of 3000.

As a result of the TER(UADUMP) LE option, and the provision of a SYSMDUMP DD statement, Fault Analyzer is invoked via the MVS change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

End of Korean feature note

**Note:** Due to differences in version, release or maintenance level of the compiler used, program offset information might differ from the sample below.

A system abend 0C9 occurred in module IDISPLI1 program IDISPLI1 at offset X'286'.

## Verifying the use of Fault Analyzer with PL/I

A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated with this abend and indicates that:

The divisor was zero in a signed binary division.

The cause of the failure was program IDISPLI1 in module IDISPLI1. The PL/I source code that immediately preceded the failure was:

```
List  
Stmt #  
-----  
000011      Array1(1) = Array1(2) / Divisor;
```

Data field values at time of abend:

```
List  
Stmt #  
-----  
000009  ARRAY1(1) FIXED BIN(31,0) AUTO = X'00000001'  
000009  ARRAY1(2) FIXED BIN(31,0) AUTO = X'00000003'  
000009  DIVISOR   FIXED BIN(31,0) AUTO = X'00000000'
```

A complete sample report from running this IVP is provided as member IDISRP02 in the IDI.SIDIDOC1 data set.

## Verifying the use of Fault Analyzer with C

This section is only applicable if you have C installed at your site.

To verify Fault Analyzer with C, edit and submit the sample job IDIVPC in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job compiles and executes a C program in IDI.SIDISAM1(IDISRC1), which abends and terminates the job step with a return code of 3000.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

### End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

### End of Korean feature note

A system abend 0C9 occurred in module G0 program IDISRC1 at offset X'45C'.

A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated with this abend and indicates that:

The divisor was zero in a signed binary division.

The cause of the failure was program IDISRC1 in module G0. The C source code

that immediately preceded the failure was:

```
Source  
File # Line #  
-----  
000000 000097      d = (a + b) / (c - 42);      /* abend */
```

where source file #:

```
000000 = system-id://'IDI.SAMPLES(IDISRC1)'
```

A complete sample report from running this IVP is provided as member IDISRP06 in the IDI.SIDIDOC1 data set.

---

## Verifying the IDIXCEE Language Environment exit enablement

If you choose to enable the IDIXCEE Language Environment exit, then the following can be used to verify that it is working correctly:

1. Do one of the following:
  - a. If using COBOL or PL/I, then edit either the IDIVPCOB or the IDIVPPLI job in data set IDI.SIDISAM1 and change the GO step LE option TER(UADUMP) to TER(TRACE).
  - b. If using assembler, then edit the IDIVPBLE job in data set IDI.SIDISAM1. This IVP is equivalent to the IDIVPASM IVP, but uses LE-compliant assembler.
2. Submit the job.
3. Go to the bottom of the IDIREPRT output and check that it includes a paragraph starting with

```
Fault Analyzer was invoked via the LE CEEEXTAN exit (IDIXCEE)...
```

in which case the IDIXCEE exit is working correctly.

---

## Verifying the IDITABD USERMOD installation

If you choose to install the IDITABD USERMOD, then the following can be used to verify that it is working correctly:

1. Edit the IDIVPASM job in data set IDI.SIDISAM1.
2. Comment out the //G.SYSUDUMP DD statement.
3. Submit the job.
4. If Fault Analyzer was invoked at all, then it must have been via the IDIXDCAP exit, and as a result of the successful installation of the IDITABD USERMOD. This exit used can be verified by going to the bottom of the IDIREPRT output and checking that it includes a paragraph starting with

```
Fault Analyzer was invoked via the MVS IEAVTABX exit (IDIXDCAP)...
```

---

## Verifying the customization of Fault Analyzer under CICS

This section is only applicable if you have CICS installed, and have completed the customization of CICS as described in Chapter 21, "Customizing the CICS environment," on page 301.

Having installed the CFA transaction, and the exits that you want, invoke the CFA transaction from a CICS terminal, and press PF9 to see the following display:

## Verifying the customization of Fault Analyzer under CICS

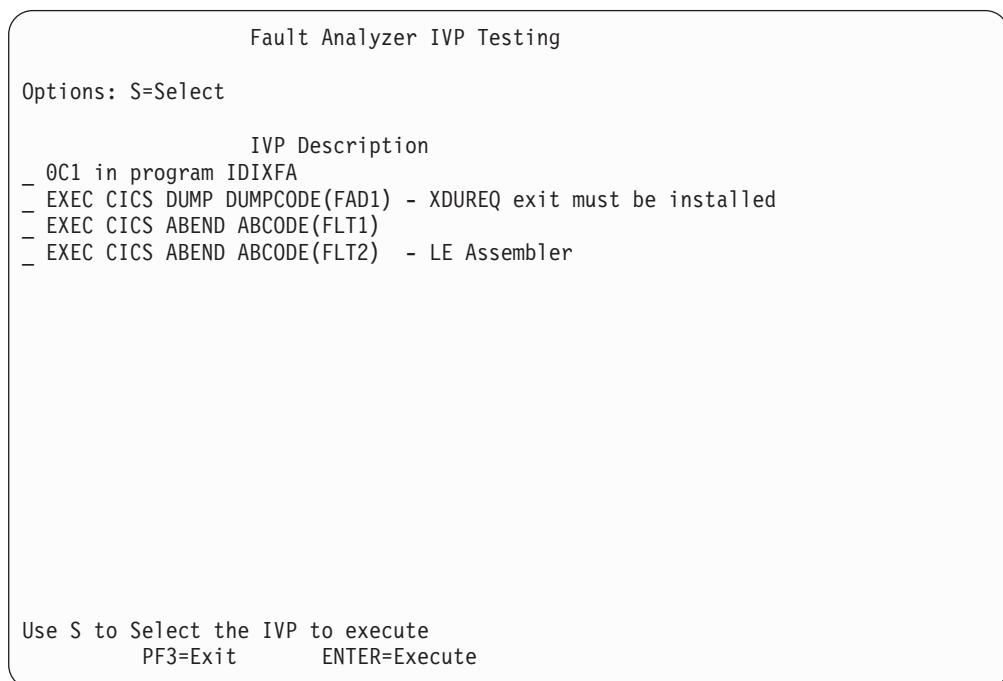


Figure 143. Sample CFA IVP Testing display

From here, type S next to the desired tests (note that the XDUREQ exit must be installed for the EXEC CICS DUMP DUMPCODE(FAD1) test), and press Enter.

Since DeferredReport is the default option for CICS, determine the history file name and fault ID for each IVP from the IDI0003I messages, and then, using the Fault Analyzer ISPF interface, issue the 'V' line command against each fault entry to view the saved report.

### CICS IVP: 0C1 in program IDIXFA

The synopsis section of the Fault Analyzer report should contain the following for the "0C1 in program IDIXFA" test:

A CICS abend ASRA occurred in module IDIXFA CSECT IDIXFA at offset X'5FC'.

A program interruption code 0001 (Operation Exception) is associated with this abend and indicates that:

An attempt was made to execute an instruction with an invalid operation code.

The abend was caused by an undetermined instruction.

NOTE: Source code information could not be presented because the search for a compiler listing or side-file was unsuccessful for CSECT IDIXFA.

### CICS IVP: EXEC CICS DUMP DUMPCODE(FAD1)

The synopsis section of the Fault Analyzer report should contain the following for the "EXEC CICS DUMP DUMPCODE(FAD1)" test:

Fault Analyzer was invoked using the EXEC CICS DUMP interface.

### CICS IVP: EXEC CICS ABEND ABCODE(FLT1)

The synopsis section of the Fault Analyzer report should contain the following for the "EXEC CICS ABEND ABCODE(FLT1)" test:

## Verifying the customization of Fault Analyzer under CICS

A CICS abend FLT1 occurred in module IDIXFA CSECT IDIXFA at offset X'666'.

The abend was caused by machine instruction 05EF (BRANCH AND LINK).

NOTE: Source code information could not be presented because the search for a compiler listing or side-file was unsuccessful for CSECT IDIXFA.

## CICS IVP: EXEC CICS ABEND ABCODE(FLT2)

The synopsis section of the Fault Analyzer report should contain the following for the "EXEC CICS ABEND ABCODE(FLT2)" test:

A CICS abend FLT2 occurred in module IDIVPCLE CSECT IDIVACLE at offset X'98'.

The abend was caused by machine instruction 05EF (BRANCH AND LINK).  
This was an EXEC CICS ABEND command.

NOTE: Source code information could not be presented because the search for a compiler listing or side-file was unsuccessful for CSECT IDIVACLE.

---

## Verifying the use of Fault Analyzer with DB2

This section is only applicable if you have DB2 installed at your site.

Two different methods are provided for verification of Fault Analyzer with DB2:

- Using a C program.

This IVP is self-contained and does not require any special setup of DB2 prior to execution. For details, see "Using a C program."

- Using a COBOL program.

This IVP requires that sample DB2 data bases have been setup prior to execution. For details, see "Using a COBOL program" on page 333.

## Using a C program

To verify Fault Analyzer with DB2 using a C program, edit and submit the sample job IDIVPDB2 in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job executes an already compiled and linked ODBC C program, which has been provided as load module IDIVPDB2 in data set IDI.SIDIAUTH. The program deliberately abends with a system abend code of S0C4.

**Note:** This IVP is based on the DB2 ODBC IVP that is usually shipped by DB2 in the DSN.SDSNSAMP data set as members DSNTSJ8 (JCL) and DSN8OIVP (C source code). This IVP has been modified to deliberately abend while in the connection with DB2 so that Fault Analyzer is invoked and includes a report section for DB2 information. The Fault Analyzer version of the source code is provided for your reference at the end of the IDIVPDB2 sample member.

As a result of the TER(UATRACE) LE option, and the provision of a SYSMDUMP DD statement, Fault Analyzer is invoked via the MVS change options/suppress dump exit, IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

## Verifying the use of Fault Analyzer with DB2

### Japanese feature note

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

### End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

### End of Korean feature note

A system abend 0C4 reason code X'4' occurred in module IDIVPDB2 program IDIVCDB2 at offset X'C74'.

A program-interruption code 0004 (Protection Exception) is associated with this abend and indicates that:

A protection exception occurred due to one of the following:

- An attempt to access a protected storage location using an incorrect storage access key.
- An attempt to store, in the access-register mode, by means of an access-list entry which has the fetch only bit set to one.
- An attempt to store into the range 0-511 or 4096-4607 with low-address protection enabled.
- An attempt to store into a protected page with DAT on.

The abend was caused by machine instruction 50000000 (STORE).

NOTE: Source code information for program IDIVCDB2 could not be presented because no compiler listing or side-file data sets were provided. The source line # from the GONUMBER option is 123 for offset X'C74'.

A complete sample report from running this IVP is provided as member IDISRP04 in the IDLSIDIDOC1 data set.

The data written to SYSPRINT should contain:

```
IDIVPDB2 INITIALIZATION
IDIVPDB2 SQLAllocEnv
IDIVPDB2-henv=1
IDIVPDB2 SQLAllocConnect
IDIVPDB2-hdbc=1
IDIVPDB2 SQLConnect
IDIVPDB2 successfully issued a SQLconnect
IDIVPDB2 SQLAllocStmt
IDIVPDB2-hstmt=1
IDIVPDB2 successfully issued a SQLAllocStmt
IDIVPDB2 SQLExecDirect
IDIVPDB2-sqlstmt=SELECT * FROM SYSIBM.SYSDUMMY1
IDIVPDB2 successfully issued a SQLExecDirect
IDIVPDB2 SQLFetch
IDIVPDB2 successfully issued a SQLFetch
IDIVPDB2 SQLTransact
IDIVPDB2 successfully issued a SQLTransact
IDIVPDB2 Abend S0C4 to invoke Fault Analyzer...
```

## Using a COBOL program

To verify Fault Analyzer with DB2 using a COBOL program, edit and submit the sample job IDIVPDBB in data set IDI.SIDISAM1. This job uses as input another sample member, IDISDB2B, containing the COBOL program source code. Refer to the instructions in the sample job for more information.

The job compiles and executes a COBOL program, which abends with a system abend code of S0C9.

**Note:** This IVP is based on the DB2 COBOL IVP that is usually shipped by DB2 in the DSN.SDSNSAMP data set as members DSNTEJ2C (JCL) and DSN8BC3 (COBOL source code). This IVP has been modified to deliberately abend after having performed DB2 data base access, so that Fault Analyzer is invoked and includes a report section for DB2 information.

Prior to running this IVP, it is important to ensure that the DB2 sample data base environment is set up correctly. This setting up is best done by following the instructions provided in the *DB2 Universal Database for OS/390 and z/OS: Installation Guide* for running the DSNTEJ2C DB2 IVP. Once DSNTEJ2C is running correctly, then either make the changes listed in the Fault Analyzer IDIVPDBB sample to the DB2 DSNTEJ2C sample, or make the same changes that were made to the DB2 DSNTEJ2C sample in the Fault Analyzer IDIVPDBB sample.

As a result of the TER(TRACE) LE option that is specified on the DB2 RUN command for the IDISDB2B program, Fault Analyzer is invoked via the LE CEEEXTAN exit, IDIXCEE.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, this section should be in Korean if the Language(KOR) option is in effect.

End of Korean feature note

**Note:** Due to differences in version, release or maintenance level of the compiler used, program offset information might differ from the sample below.

A system abend 0C9 occurred in module IDISDB2B program IDISDB2B at offset X'1EE2'.

A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated with this abend and indicates that:

The divisor was zero in a signed binary division.

The cause of the failure was program IDISDB2B in module IDISDB2B. The COBOL source code that immediately preceded the failure was:

## Verifying the use of Fault Analyzer with DB2

```
Source  
Line #  
-----  
001165      DIVIDE NOT-FOUND BY PERCENT-COUNTER  
001166      GIVING ERROR-TEXT-LEN.
```

The COBOL source code for data fields involved in the failure:

```
Source  
Line #  
-----  
000137      77  NOT-FOUND          PIC S9(9) COMP VALUE +100.  
000146      77  ERROR-TEXT-LEN    PIC S9(9) COMP VALUE +120.  
000207      *  
000208      77  PERCENT-COUNTER   PIC S9(4) COMP.
```

Data field values at time of abend:

```
ERROR-TEXT-LEN = 120  
NOT-FOUND      = 1  
PERCENT-COUNTER = 0 *** Cause of error ***
```

The analysis should include a DB2 Information section similar to the following:

**Note:** Installation-specific names and values are likely to differ from those shown in the sample below.

---

### <H3> DB2 Subsystem DB42

```
DB2 Version . . . . . : V8R1M5  
Plan Name . . . . . : DSN8BH81 (Bound 2006/08/25 14:30:52)  
Plan Owner. . . . . : SWILKEN  
Database Request Module Name: DB2V810.DB42.DBRLIB.DATA(IDISDB2B)  
Consistency Token . . . . : X'17E9C40018AE6A18'  
Primary Authorization ID. . . : SWILKEN  
Current SQL ID. . . . . : SWILKEN
```

```
Source  
Line #  
-----  
Last Executed SQL Statement : 001149      *****      EXEC SQL FETCH TELE1 INTO :PPHONE END-EXEC.  
  
Fault Analyzer Event #. . . . : 4 (Program IDISDB2B)  
Declare Cursor Stmt No. . . . : 200  
Declare Cursor Stmt . . . . : DECLARE TELE1 CURSOR FOR SELECT * FROM DSN8810 .  
                           VPHONE  
Open Cursor Stmt No . . . . : 346  
Open Cursor Stmt. . . . . : OPEN TELE1
```

Output Host Variables:

```
Name and Data Type. . . . . : PPHONE.LASTNAME VARCHAR(15)  
At Address. . . . . : 168A83D8  
Data Value. . . . . : HAAS
```

```
Name and Data Type. . . . . : PPHONE.FIRSTNAME VARCHAR(12)  
At Address. . . . . : 168A83E9  
Data Value. . . . . : CHRISTINE
```

```
Name and Data Type. . . . . : PPHONE.MIDDLEINITIAL CHARACTER(1)  
At Address. . . . . : 168A83F7  
Data Value. . . . . : I
```

```
Name and Data Type. . . . . : PPHONE.PHONENUMBER CHARACTER(4)
At Address. . . . . : 168A83F8
Data Value. . . . . : 3978
```

```
Name and Data Type. . . . . : PPHONE.EMPLOYEEENUMBER CHARACTER(6)
At Address. . . . . : 168A83FC
Data Value. . . . . : 000010
```

```
Name and Data Type. . . . . : PPHONE.DEPTNUMBER CHARACTER(3)
At Address. . . . . : 168A8402
Data Value. . . . . : A00
```

```
Name and Data Type. . . . . : PPHONE.DEPTNAME VARCHAR(36)
At Address. . . . . : 168A8405
Data Value. . . . . : SPIFFY COMPUTER SERVICE DIV.
```

---

### <H3> DB2 Control Blocks

SQL Communications Area (SQLCA) for subsystem DB42 not shown as it is identical to the SQLCA in the detail section for event # 4 program IDISDB2B.

---

## Verifying the use of Fault Analyzer through ISPF

To verify Fault Analyzer under ISPF, invoke the Fault Analyzer ISPF interface through the ISPF option that you set up in Chapter 15, “Modifying your ISPF environment,” on page 247. This invocation should present the Fault Entry List display, containing entries for the abends that occurred as a result of submitting the verification programs.

Initially, the default history file IDI.HIST, or the history file that is specified in the IDICNF00 parmlib member DataSets option, is used for the display.

To inspect the Fault Analyzer report generated at the time of abend, enter the command 'V' next to the entry you want to view, and press Enter.

For more information about how to use the Fault Analyzer ISPF interface, refer to Chapter 3, “The Fault Analyzer ISPF interface,” on page 33.

---

## Verifying the recovery fault recording set-up

To verify the set-up of recovery fault recording, insert a JCL statement like the following into the abending step of any of the other IVP jobs, for example, the IDIVPCOB IVP job (see “Verifying the use of Fault Analyzer with COBOL” on page 326):

```
//GO.IDIRFRON DD DUMMY
```

When the IDIRFRON DDname is allocated to the abending step being analyzed by Fault Analyzer, then a deliberate abend U0777 is issued. This abend activates the recovery fault recording feature to write a IEATDUMP with the name as specified using the IDISRFR usermod (or the default name, if the usermod has not been installed), and a recovery fault recording fault entry.

Before submitting the job, ensure that the IDIS subsystem is started.

Messages, similar to the following, should be expected:

## Verifying the recovery fault recording set-up

```
+IDI0001I Fault Analyzer V9R1M0 (MVS 2008/12/10) invoked by IDIXCEE using SYS1.PARMLIB.FAE1.USER(IDICNF00)
+IDI0047S IBM Fault Analyzer internal abend. U0777
+IDI0126I Recovery fault recording fault ID BAT15874 assigned in history file DA.DCAT
IGD101I SMS ALLOCATED TO DDNAME (SYS00038) 524
      DSN (IDIRFRHQ.IDIRFR.FAE1.D081215.T013821.IDIVPCO)
      STORCLAS (SCIDIRFR) MGMTCLAS (PRIMARY) DATACLAS (DEFAULT)
      VOL SER NOS= E$RF01
IGD104I IDIRFRHQ.IDIRFR.FAE1.D081215.T013821.IDIVPCO RETAINED, DDNAME=SYS00038
IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO IDIRFRHQ.IDIRFR.FAE1.D081215.T013821.IDIVPCO
```

Reanalysis of the recovery fault recording fault entry identified in the IDI0126I message should result in a report, which is identical to the one from the same IVP run without the IDIRFRON DD statement.

For information about the recovery fault recording data set name being used, see “Changing the default recovery fault recording IEATDUMP data set name” on page 258.

---

## Chapter 27. Managing history files (IDIUTIL utility)

A number of different functions are required to help manage fault history files. The most obvious of these is the ability in a batch utility to delete a set of entries based on some criteria such as date, in order to keep the number of entries in the history file at a manageable level.

Such a utility program is provided with Fault Analyzer as a load module named IDIUTIL. It is a batch history file utility that can be used to perform history file management functions, such as listing and deleting history file fault entries.

**Note:** Do not delete members from a history file PDS(E) outside of the Fault Analyzer ISPF interface or the IDIUTIL batch utility (for example, directly from an ISPF data set member list). If you do, the history file index is out of synch until the recording of the next fault during real-time analysis or a run of the IDIUTIL batch utility against the history file.

There is also some value in being able to list the entries in history files in order to keep track of problems.

The maintenance functions of IDIUTIL are driven by a series of control statements that it reads from the SYSIN DD JCL data set. The control statements start in column 1 of the SYSIN records with any continuation statements having a blank in column 1. Comments can be placed in this control statement stream with an asterisk in column 1 of the comment line.

The SYSIN stream is processed sequentially, one control statement at a time. The target history files for control statements can be implicit or explicit depending on the control statement. The control statements that set up target history file data set names overwrite the target history file names previously in effect. The FILES control statement's only purpose is to set the target history file data set names for following control statements. This purpose makes sense when you see that the LISTHF and DELETE control statement syntax does not include a target history file keyword. They operate on the current target history file set, which can be one or more data set names.

Other control statements such as IMPORT and SETFAULTPREFIX carry a single history file in their syntax which resets the current history file set to that data set name, before carrying out its action.

As an alternative to the SYSIN stream, control statements for the IDIUTIL batch utility can be passed via the EXEC JCL statement PARM field. Control statements passed in this way must not include any imbedded blank spaces, but must be separated from other control statements by one or more blanks.

The intention of the IDIUTIL batch utility is to provide a base set of capabilities that include selection of fault entries to list or delete based on criteria such as date and job name. In order to provide for more advanced requirements there are three user exit points that permit more advanced selection and recording to be coded by the user. These exits are for the DELETE, LISTHF, and IMPORT control statements. They follow the same structure as the user exits provided with the Fault Analyzer real time and reanalysis functions. They can be written in REXX, Assembler, or a high-level language.

## **IDIUTIL control statements**

The control statements are presented here in the order that helps explain their use, but the only order requirement for execution is that the LISTHF and DELETE control statements need to be preceded by one of the other control statements that populate the history file data set name set. The FILES control statement is the only one that can put more than one data set name into the set of history file data set names. All of the other control statements (apart from LISTHF, DELETE, and Exits) reset the current data set names to a single target data set name.

In the following syntax diagrams, either commas or blank characters are permitted as delimiters when repeating suboptions or values.

### **FILES control statement**

#### **Syntax**



#### **Description**

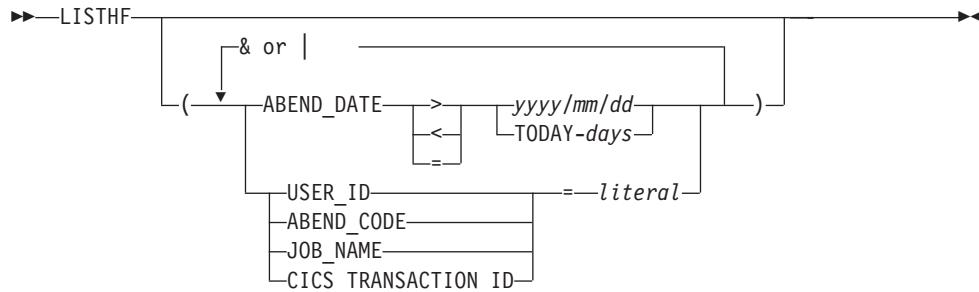
The FILES control statement is a single keyword followed by a set of parentheses containing PDS(E) history file data set names.

The LISTHF and DELETE control statements immediately following the FILES control statement operate on all of the history files in the FILES statement.

Examples showing the use of the FILES control statement are provided in "Examples" on page 344.

### **LISTHF control statement**

#### **Syntax**



## Description

The LISTHF (LIST History File) control statement is a single keyword, optionally followed by parentheses containing qualifiers to select which fault entries should be listed.

The qualifiers have a basic capacity to compare greater than, less than, or equal for the fault entry ABEND\_DATE, USER\_ID, ABEND\_CODE, JOB\_NAME, or CICS\_TRANSACTION\_ID (all of which are field names in the ENV data area) to a literal in the LISTHF control statement. Comparisons can be combined with and, or (& |) operators. The result of this simple syntax capability can be passed on to a user exit if more complex comparisons are desired.

Two special literal comparison qualifiers are recognized. An asterisk in the literal truncates the comparison for wild card capabilities, such as:

`JOB_NAME = AB*`

The other special literal is TODAY-*days*, which is converted to today's date, minus the number of numeric days specified at *days*, and then converted to a string of the 2001/02/23 format before comparison. Naturally, the TODAY-*days* literal is only meaningful when used with ABEND\_DATE, such as:

`ABEND_DATE < TODAY-30`

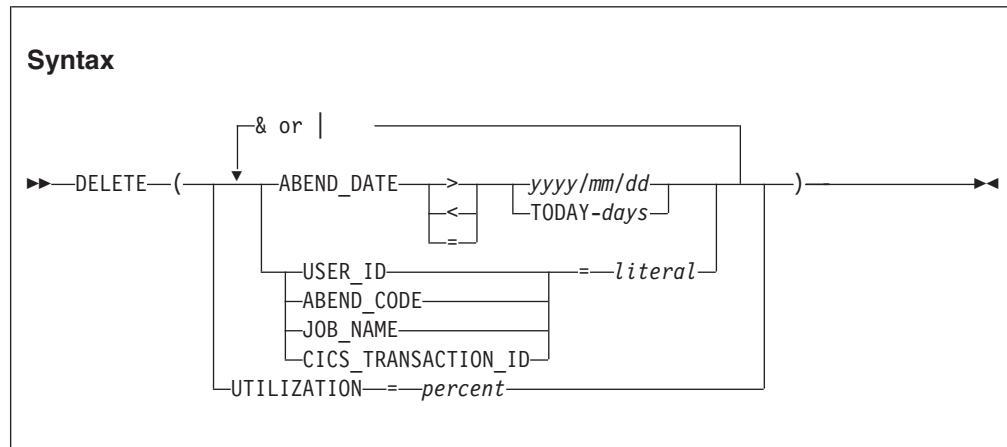
The value specified for *days* must be between 0 and 2147483647, both inclusive.

When comparing ABEND\_CODE, the format is four numeric digits for user abend codes and three hex digits preceded by S for system abends. For example, S0C4 for a system 0C4 and 4038 for a user 4038 abend. CICS abend codes are four alphabetic characters, for example, ASRA.

The IDIUTIL ListHF user exit (see “IDIUTIL ListHF user exit” on page 405) can be used with the LISTHF control statement to apply extra selection criteria to the history file entries that should be listed.

An example showing the use of the LISTHF control statement is provided in “Example 1. Listing history file entries” on page 344.

## DELETE control statement



## IDIUTIL control statements

### Description

The DELETE control statement is a single keyword followed by parentheses containing qualifiers to select which fault entries should be deleted.

The IDIUTIL DELETE function should only be used for PDS history files. PDSE history files (which we recommend are used) should use automatic space management. For information about automatic space management, see “AUTO-managed PDSE history files” on page 262.

**Note:** Deletion of a locked fault entry is only possible when overriding the default action using an IDIUTIL Delete user exit. For more information about the IDIUTIL Delete user exit specification and usage, see “Description” on page 343. For general information about the lock flag, and how to change its value prior to running the IDIUTIL batch utility, see “Viewing fault entry information” on page 82.

The UTILIZATION qualifier can be used to delete entries, starting with the oldest entry, until the specified percentage of utilization is reached. This qualifier is only available for PDSE history files. UTILIZATION cannot be qualified with extra & or | operators, it must be the only operator in the DELETE statement when used. It can be preceded or followed by any other DELETE statements.

**Note:** When using the UTILIZATION qualifier, all IDIUTIL does is try to delete usage back to the specified percentage full at the time of running IDIUTIL. It is *not* a way of telling Fault Analyzer to maintain the usage at the specified percentage as new fault entries are created. PDSE history files are by default AUTO-managed and use the existing data set extents as much as possible, without allocating more extents.

The remaining qualifiers follow the same rules as the LISTHF qualifiers above.

The IDIUTIL Delete user exit (see “IDIUTIL Delete user exit” on page 404) can be used with the DELETE control statement to further select the history file entries that should be deleted.

Examples showing the use of the DELETE control statement are provided in “Example 2. Deleting history file entries by date” on page 344 and “Example 3. Deleting history file entries by utilization” on page 345.

## SETFAULTPREFIX control statement

### Syntax

```
►►SETFAULTPREFIX—(—data-set-name—,—prefix—)————►►
```

### Description

The SETFAULTPREFIX control statement is a single keyword followed by a set of parentheses containing the PDS(E) history file data set name whose fault entry prefix characters are changed, followed by up to three alphabetic characters which become the new prefix characters. Only alphabetic prefix characters are permitted.

Prefixes for all existing fault IDs in the specified history file are changed to the specified prefix, and all fault IDs later created in this history file receive this prefix automatically.

**Note:** By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in “Restricting change of history file settings” on page 233.

An example showing the use of the SETFAULTPREFIX control statement is provided in “Example 4. Changing history file fault prefix characters” on page 345.

## **SETMAXFAULTENTRIES control statement**

### **Syntax**

```
►—SETMAXFAULTENTRIES—(—data-set-name—,—max-number—)—————►
```

### **Description**

The SetMaxFaultEntries control statement is a single keyword, followed by a set of parentheses containing a PDS history file data set name and the maximum number of fault entries to be set.

*max-number* specifies the maximum number of fault entries that can be maintained in the history file. Additional data set extents are allocated as needed to achieve this number of fault entries. An out-of-space condition occurs if there is no space available in the data set (that is, the number of data set extents has reached the maximum, or the volume is full) prior to the history file containing *max-number* fault entries.

*max-number* must be 1 - 2147483647.

**Note:** If the maximum number of fault entries specified exceeds the current number of fault entries in the history file, then the oldest fault entries that are not locked are deleted until the history file only contains the number of fault entries specified.

This option is available for PDS history files only—use SetMinFaultEntries for PDSE history files.

**Note:** By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in “Restricting change of history file settings” on page 233.

## **SETMINFAULTENTRIES control statement**

### **SETMINFAULTENTRIES control statement**

#### **Syntax**

```
►►—SETMINFAULTENTRIES—(—data-set-name—,—min-number—)————►►
```

#### **Description**

The SetMinFaultEntries control statement is a single keyword, followed by a set of parentheses containing a PDSE history file data set name and the minimum number of fault entries to be set.

The history file is maintained automatically once a minimum of *min-number* fault entries exist in the history file, regardless of how many data set extents have been allocated to achieve this. Once the history file is maintained automatically, then the number of fault entries is limited only by the currently available data set space. No extra data set extents are generally allocated, and no out-of-space conditions are expected.

This option is available for PDSE history files only—use SetMaxFaultEntries for PDS history files.

See “AUTO-managed PDSE history files” on page 262 for more information about AUTO-managed PDSE history files.

*min-number* must be 25 - 2147483647.

**Note:** By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in “Restricting change of history file settings” on page 233.

An example showing the use of the SetMinFaultEntries control statement is provided in “Example 5. Creating a self-maintained history file” on page 345.

## **IMPORT control statement**

#### **Syntax**

```
►►—IMPORT—(—to-data-set-name—,—from-data-set-name—[—faultid—])————►►
```

#### **Description**

The IMPORT control statement is a single keyword followed by a set of parentheses containing the "to" PDS(E) history file data set name followed by the "from" data set name.

The fault entries in the "from" data set are copied and their prefix characters set to the prefix of the "to" history file. When possible, the imported fault number is left the same, however, the fault number is altered where necessary to ensure no existing "to" history file fault entries are overwritten. After successful copy, the fault member is deleted in the "from" data set.

If a single fault ID is included with the "from" data set specification, for example TEMP.HIST(F00234), then only that fault ID is imported and deleted.

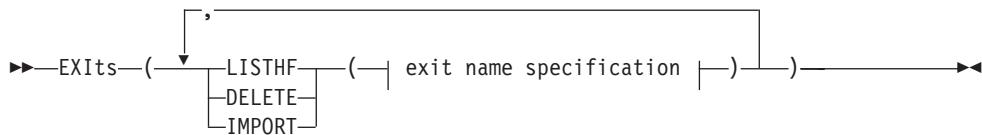
This function allows fault members that have been XMIT'ed from one system to be received into a holding data set, and then IMPORT'ed to the required target history file. For an example of this, refer to "Managing history files across MVS systems without shared DASD" on page 266. The use of the IMPORT user exit with this function could allow users to be sent notification that the imported entries have arrived.

The IDIUTIL Import user exit (see "IDIUTIL Import user exit" on page 403) can be used with the IMPORT control statement to further select the history file entries that should be imported.

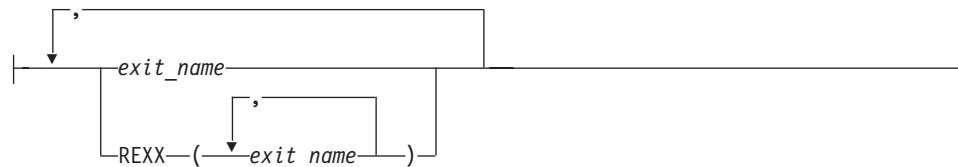
An example showing the use of the IMPORT control statement is provided in "Example 6. Importing history file entries" on page 346.

## EXITS control statement

### Syntax



### exit name specification:



### Description

The Exits control statement follows the same format as the Exits keyword for real-time analysis and reanalysis. The difference is that the exit points are for LISTHF, DELETE, and IMPORT.

The exit is driven for every fault entry in the LISTHF or DELETE target data sets that match the specified selection qualifiers, and for the members found in the 'from' data set for IMPORT. In all cases, the UTL.PERFORM\_ACTION flag is set to 'Y' by default, except when an IDIUTIL Delete user exit is called for a locked fault

## **EXITS control statement**

entry. In this case, when ENV.LOCK\_FLAG is not blank, the UTL.PERFORM\_ACTION flag is set to 'N' before passing control to the user exit.

The Exits control statement remains in effect for any LISTHF, DELETE, or IMPORT control statements that follow, or until a new Exits control statement is encountered for this run of the utility. The effect of multiple Exits control statements is not cumulative—the previous exits are cleared on encountering a new Exits control statement. There are no initial user exits active at the start of IDIUTIL execution and the LISTHF, DELETE, and IMPORT exit points are not recognized in or read from the configuration files used by Fault Analyzer real time analysis and reanalysis.

Deprecated options ACCOUNTING and NOACCOUNTING can still be specified, but are ignored.

A detailed description of each exit type is provided in Chapter 31, “Customizing Fault Analyzer by using user exits,” on page 359.

An example showing the use of the Exits control statement is provided in “Example 6. Importing history file entries” on page 346.

---

## **Examples**

The following are examples showing the use of the IDIUTIL batch utility.

### **Example 1. Listing history file entries**

This example shows an IDIUTIL batch utility job to list all history file entries that are contained in the history file MY.HIST1 and all entries in history file MY.HIST2 that are for job names starting with TEMP, contain an abend code of S0C1, and are for user ID P0001.

```
//UTILJOB1 JOB ...
//RUNUTIL EXEC PGM=IDIUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
* In the first history file, list all entries
FILES(MY.HIST1)
LISTHF
* In the second history file, only list those entries that match a specific criteria
FILES(MY.HIST2)
LISTHF(USER_ID=P0001 & ABEND_CODE=S0C1 &
      JOB_NAME=TEMP*)
/*
```

### **Example 2. Deleting history file entries by date**

This example shows an IDIUTIL batch utility job to delete all history file entries in the history files MY.HIST1 and MY.HIST2 that are more than two weeks old.

```
//UTILJOB2 JOB ...
//RUNUTIL EXEC PGM=IDIUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
FILES(MY.HIST1,MY.HIST2)
DELETE(ABEND_DATE < TODAY-14)
/*
```

### Example 3. Deleting history file entries by utilization

This example shows an IDIUTIL batch utility job to delete history file entries in the history files MY.HIST1 and MY.HIST2, until the history file utilization is less than 80 percent. The oldest entries are deleted first.

```
//UTILJOB2 JOB ...
//RUNUTIL EXEC PGM=IDIUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
FILE(S(MY.HIST1,MY.HIST2)
DELETE(UTILIZATION = 80)
/*
```

Note that the history file must be a PDSE in order to perform this function.

### Example 4. Changing history file fault prefix characters

This example shows an IDIUTIL batch utility job to change the fault prefix characters for history file MY.HIST to ABC.

```
//UTILJOB3 JOB ...
//RUNUTIL EXEC PGM=IDIUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SETFAULTPREFIX(MY.HIST,ABC)
/*
```

### Example 5. Creating a self-maintained history file

This example shows a job to allocate a PDSE history file that is named MY.HIST, and using the IDIUTIL batch utility, set the fault wrap number for to 100 while permitting the future reuse of existing fault numbers.

```
//UTILJOB4 JOB ...
//IDCAMS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    ALLOC DSNAME('MY.HIST')      -
        NEW                  -
        SPACE(10 10)          -
        CYLINDERS            -
        RECFM(V B)           -
        LRECL(10000)          -
        DIR(10)               -
        DSNTYPE(LIBRARY)      -
/*
//RUNUTIL EXEC PGM=IDIUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SetMinFaultEntries(MY.HIST,100)
/*
```

This code creates a history file with maximum automatic free space reclamation that should never run out of space provided that the data set allocation is sufficient to hold the minimum number of faults as per the SetMinFaultEntries specification.

You might also want to add a SETFAULTPREFIX control statement to your history file creation jobs (see “SETFAULTPREFIX control statement” on page 340) to make fault IDs in each new history file unique.

## Examples

### Example 6. Importing history file entries

This example shows an IDIUTIL batch utility job to import all history file entries from MY.TEMP.HIST to MY.HIST that occurred on system name CICS04. Because of the need to test for system name in this example, an IDIUTIL Import user exit is required.

Assuming that MY.TEMP.HIST contains the faults:

Fault ID	Job/Tran	User	System	Abend	Date	Time
F00101	IDIVPCOB	SWILKEN	MVS2	S0C7	2001/10/18	08:57:08
F00098	FRED	CICSUSER	CICS02	AEIL	2001/10/15	14:33:30
F00097	WILMA	CICSUSER	CICS04	AEIL	2001/10/15	13:00:57
F00096	BARNEY	CICSUSER	CICS02	AEIL	2001/10/15	12:56:32
F00095	BUSHBY2N	SWILKEN	MVS2	U4038	2001/10/14	10:41:29
F00093	BETTY	CICSUSER	CICS04	ASRA	2001/10/12	21:16:37
F00092	DACBB045	SWILKEN	MVS2	U4038	2001/10/10	10:38:22

and MY.HIST contains the faults:

Fault ID	Job/Tran	User	System	Abend	Date	Time
F00030	BUSHBY2A	BUSHBYD	MVS2	U4038	2001/09/18	13:02:02
F00060	IMSLE4	SWILKEN	MVS1	S0C9	2001/09/12	12:39:27
F00059	IMSLE3	SWILKEN	MVS2	U4036	2001/09/12	12:38:31

then running the JCL:

```
//UTILJOB5 JOB ...
//RUNUTIL EXEC PGM=IDIUTIL
//IDIEEXEC DD DISP=SHR,DSN=MY.REXX.EXECS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
Exits(IMPORT(REXX(IMPXMP)))
IMPORT(MY.HIST,MY.TEMP.HIST)
/*
```

with the IDIUTIL Import user exit in member IMPXMP of data set MY.REXX.Exits:

```
/* REXX */
If ENV.SYSTEM_NAME ~= 'CICS04' then UTL.PERFORM_ACTION = 'N'
```

results in these fault entries in MY.TEMP.HIST:

Fault ID	Job/Tran	User	System	Abend	Date	Time
F00101	IDIVPCOB	SWILKEN	MVS2	S0C7	2001/10/18	08:57:08
F00098	FRED	CICSUSER	CICS02	AEIL	2001/10/15	14:33:30
F00096	BARNEY	CICSUSER	CICS02	AEIL	2001/10/15	12:56:32
F00095	BUSHBY2N	SWILKEN	MVS2	U4038	2001/10/14	10:41:29
F00092	DACBB045	SWILKEN	MVS2	U4038	2001/10/10	10:38:22

and these fault entries in MY.HIST:

Fault ID	Job/Tran	User	System	Abend	Date	Time
F00030	BUSHBY2A	BUSHBYD	MVS2	U4038	2001/09/18	13:02:02
F00031	WILMA	CICSUSER	CICS04	AEIL	2001/10/15	13:00:57
F00032	BETTY	CICSUSER	CICS04	ASRA	2001/10/12	21:16:37
F00060	IMSLE4	SWILKEN	MVS1	S0C9	2001/09/12	12:39:27
F00059	IMSLE3	SWILKEN	MVS2	U4036	2001/09/12	12:38:31

Note that the entries that were imported into MY.HIST have been deleted from MY.TEMP.HIST.

An extra example showing the use of the IDIUTIL batch utility import function is shown in "Managing history files across MVS systems without shared DASD" on page 266.

## **IDIUTIL batch utility user exit samples**

User exit samples for the IDIUTIL batch utility can be found in “Descriptions of IDIUTIL batch utility user exit types” on page 403.

In particular, the sample shown in “Example 2” on page 407 illustrates how a customized report might be written, and a comma-delimited file generated, which can be used as input to a spreadsheet application.



---

## Chapter 28. Providing application-specific explanations and descriptions

The member IDIHUSR M of the IDI.SIDIDOC1 data set holds user-defined message and abend code explanations, as well as program name descriptions.

The structure of IDIHUSR M is straight-forward. Each item (message explanation, abend code explanation or program description) is identified by a header record, followed by the text to be displayed for that item. The text can be in more than one record. The item is finished by a new header record or the end of the file. Each record is a string. Use a text editor to maintain the items.

**Note:** If the Fault Analyzer IDIS subsystem is used, then it is necessary to stop and restart the IDIS subsystem in order to include new or changed explanations or descriptions in the in-storage cache. If the restart is not done, then new or changed explanations or descriptions are not found.

If you begin a line in IDIHUSR M with .\*, then Fault Analyzer takes it as a comment and disregards it. It is not included in the analysis report.

---

### User-defined message explanations

If one of your applications writes a message to the SYSLOG, and the application subsequently abends, then Fault Analyzer looks through IDIHUSR M, to see if it can find a message associated with the identifier of the issued message. If it finds a match, then Fault Analyzer is able to include the message explanation in the analysis report.

The message explanation header record in IDIHUSR M has the following format:

#### Syntax

►—:msg.message\_identifier—►

where

*message\_identifier*

A code identifying the message. This code can be any combination of numbers and letters. It is not case-sensitive. For example:

MYMESSAGE  
m103a  
dbg303

are all valid message identifiers.

For the purpose of providing message explanations, Fault Analyzer treats all characters from the start of any message, up until the first blank character, as constituting the message identifier being searched for.

Any plus signs (+) that MVS might add to the start of messages when these are being displayed are not considered part of the message identifier. If these are included in the IDIHUSR M member, then they are ignored.

## User-defined message explanations

Note that “:msg.” must begin in column 1.

The lines of the message following the header record are transferred directly as you type them in. So if you include leading spaces in the explanation, then Fault Analyzer includes the leading spaces when it prints the message.

In the following example, two messages are defined:

```
:msg.payrollmsg1  
Processing of accumulated leave about to commence.  
- If processing fails after this point,  
  check for negative accumulations.  
:msg.payrollmsg2  
Processing of accumulated leave completed.
```

When using the LOOKUP command, user-defined messages appear in the "Messages" main category, under the subcategory name consisting of the first 3 characters of the message ID.

---

## User-defined abend code explanations

The abend code explanation header record in IDIHUSRML has the following format:

### Syntax

►►:abend.*abend\_code*►►

where

*abend\_code*

A three-character prefix followed by a four-digit user abend code. It is not case-sensitive. For example:

```
XYZ0001  
prd1234  
zzz0999
```

are all valid abend code specifications.

The three-character abend code prefix is used to identify the load module name in which the user abend is issued. That is, the first 3 characters of the load module name in which the abend occurred, must match the specified abend code prefix. This match permits the same user abend code to be issued from different load modules, each potentially with its own unique explanation.

Note that “:abend.” must begin in column 1.

In the following example, two user abends are defined. The U1888 abend might be issued by a load module whose name begins with the 3 characters MD1, for example MD100P. The U0016 abend might be issued from a load module name whose name begins with the 3 characters EXT, for example EXTMAIN. The IDIHUSRML specification follows:

```
:abend.MD1888  
An error occurred when attempting to write the invoice  
to DDname SYSPRINT.  
:abend.EXT0016  
Incorrect EXTRACT option specified.
```

The problem might be one of the following:  
- Invalid range  
- Typo  
Respecify and try again.

When using the LOOKUP command, user-defined abend code explanations appear in the "Abend Codes" main category, under the subcategory "Other Abend Codes", and further separated into categories according to the three character load module name prefix.

## User-defined program descriptions

The program description header record in IDIHUSR.M has the following format:

### Syntax

```
►--:misc.PROGDESC:program_name--►
```

where

*program\_name*

The name of a user application entry point, program or load module. It is not case-sensitive. For example:

```
XYZMAIN  
prd01  
zzzpayr1
```

are all valid specifications.

A matching entry point name is searched for first. If a matching entry point name is not found, then the program name is searched for next. If a matching program name is not found either, then the load module name is searched for.

If a matching entry point, program or load module name is found, then its description is shown in the Fault Analyzer report Event Summary under the "Description" heading, as well as in the detail section for the event.

Note that ":misc." must begin in column 1.

In the following example, two program descriptions are specified:

```
:misc.PROGDESC:accrevb1  
Accounts receivable main module  
:misc.PROGDESC:errrrtn  
Common error routine
```

When using the LOOKUP command, user-defined program descriptions appear in the "Miscellaneous Information" main category, under the subcategory "Program Descriptions".



---

## Chapter 29. Maintaining Fault Analyzer

Fault Analyzer is maintained using the standard IBM APAR/PTF service.

++APAR fixtests need never be restored, since they will either be superseded by a PTF, or a subsequent ++APAR fixtest will specify a later SMP/E REWORK date. However, it might be necessary to perform SMP/E restore and reapply of USERMODs if the maintenance causes a conflict.

Whenever maintenance has been applied to Fault Analyzer, the following steps must be performed:

1. If Fault Analyzer SMP/E target libraries, or the target libraries of any USERMODs provided by Fault Analyzer that update libraries in other products, are in LINKLIST, then these should be removed from LLA and VLF control before performing the SMP/E APPLY. This removal is to prevent errors when attempting to load modules from LINKLIST, due to SMP/E having compressed or added extents to the libraries.
2. Do one of the following:
  - a. IPL with CLPA
  - b. Alternatively, perform dynamic updates as follows:
    - 1) If Fault Analyzer modules have been placed in LPA using the SETPROG command, as opposed to placing IDI.SIDILPA1 in LPA, then do the following:
      - a) Issue the command

```
SETPROG LPA,DELETE,MOD=(IDIDA,IDLNGX),FORCE=YES  
(For complete information on the use of the SETPROG command,  
refer to MVS System Commands.)
```
      - b) Issue the command

```
F LLA,REFRESH
```

 Optionally, to add the modules to LPA again to regain the region size space advantage, issue the command

```
SETPROG LPA,ADD,MOD=(IDIDA,IDLNGX),DSN=LNKLST
```
    - 2) If IDI.SIDILPA1 is included in your LPALIST, then issue the command

```
SETPROG LPA,ADD,MOD=(IDIDA,IDLNGX),DSN=LNKLST
```
    - 3) If using CICS, then refresh all installed CICS exits. This refresh can be done by first uninstalling, and then reinstalling the exits, using the CFA transaction as described in "Controlling CICS transaction abend analysis" on page 305.
    - 4) If using the Fault Analyzer IDIS subsystem, then stop and restart the subsystem as described in Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239.
    - 5) Users of the Fault Analyzer ISPF interface should exit and reenter ISPF for any updates to take effect.

See "Sharing of history files across a sysplex with mixed levels of Fault Analyzer" on page 216 for information about compatibility if sharing history files between sysplex images with a mix of Fault Analyzer versions or maintenance levels installed.



---

## Chapter 30. Disabling Fault Analyzer

Different options are available for disabling real-time invocation of Fault Analyzer at the system or job level, as explained in the following sections.

Regardless of the method used, SMF type 89 records are also prevented from being written.

---

### Temporarily uninstalling Fault Analyzer

If for any reason you want to temporarily uninstall Fault Analyzer, then the easiest way is to do the following:

1. Rename the following invocation exit load modules in data set IDI.SIDIAUTH, for example, by changing the first character 'T' to an 'O':  
IDIXDCAP  
IDIXTSEL  
IDIXCCEE  
IDIXCEE  
IDIXCX52  
IDIXCX53
2. Issue the MVS operator command  
F LLA,REFRESH
3. If using CICS, then either bounce all CICS systems, or use the Fault Analyzer-provided CFA transaction to uninstall all installed CICS invocation exits.

Having done the above, then the exit processes are not able to find the expected load modules, and processing continues without them.

**Note:** Remember to rename the load modules back to their original names before applying any maintenance.

**Reinstallation:** To reinstall Fault Analyzer, do the following:

1. Rename the load modules in step 1 of the above procedure back to their original names.
2. Issue the MVS operator command  
F LLA,REFRESH
3. If using CICS, then either bounce all CICS systems, or use the Fault Analyzer-provided CFA transaction to install all required CICS invocation exits.

---

### Turning off Fault Analyzer using the IFAPRDxx parmlib member

If it might be necessary to disable Fault Analyzer from running in a particular z/OS image, then this disablement can be achieved by adding an entry to your IFAPRDxx parmlib member.

If you purchased Fault Analyzer as part of product code 5655-PDS, IBM Problem Determination Solution Pack for z/OS, then modify the entry for product code 5655-PDS as follows:

## Turning off Fault Analyzer using the IFAPRDxx parmlib member

```
PRODUCT OWNER('IBM CORP')
  NAME('IBM PD SOLTN PAC')
  ID(5655-PDS)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME('PROB-DET-SOL-PAC')
  STATE(DISABLED)
```

If you purchased Fault Analyzer as part of product code 5655-DMS, IBM Problem Determination Modernization Solution Pack for z/OS, then modify the entry for product code 5655-DMS as follows:

```
PRODUCT OWNER('IBM CORP')
  NAME('IBM PD MD SO PAC')
  ID(5655-DMS)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME('PR-DE-MD-SOL-PAC')
  STATE(DISABLED)
```

If you purchased Fault Analyzer as part of product code 5697-CDT, IBM COBOL DevOps Tools Suite for z/OS V1.1, then modify the entry for product code 5697-CDT as follows:

```
PRODUCT OWNER('IBM CORP')
  NAME('IBM COB DEVOP TS')
  ID(5697-CDT)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME('FAULT-ANALYZER')
  STATE(DISABLED)
```

If you purchased Fault Analyzer as product code 5655-Q11, then modify the entry for product code 5655-Q11 as follows:

```
PRODUCT OWNER('IBM CORP')
  NAME('FAULT ANALYZER')
  ID(5655-Q11)
  VERSION(*) RELEASE(*) MOD(*)
  FEATURENAME('FAULT ANALYZER')
  STATE(DISABLED)
```

All parameters except VERSION, RELEASE and MOD must be specified exactly as shown. Any syntactically valid values can be specified for VERSION, RELEASE and MOD.

Refer to *MVS Initialization and Tuning Reference* for general information about the IFAPRDxx parmlib member.

If at a later stage Fault Analyzer needs to be reenabled, then either remove the above entry, or change the STATE to ENABLED.

---

## Turning off Fault Analyzer with a JCL switch (IDIOFF)

The Fault Analyzer invocation exits can be turned off at a job-step level by coding a

```
//IDIOFF DD DUMMY
```

JCL statement in the job step.

Using the JCL switch is more efficient than coding

```
//IDIOPTS DD *
  Exclude
/*
```

## Turning off Fault Analyzer with a JCL switch (IDIOFF)

because the JCL switch is processed much earlier by Fault Analyzer than the Exclude option (for details, see “Real-time exclusion processing” on page 27).

---

## Turning off Fault Analyzer using an environment variable (\_IDI\_OFF)

In a z/OS Unix System Services environment, the Fault Analyzer invocation through the Language Environment abnormal termination exit, IDIXCEE, can be turned off by creating an environment variable with the name

\_IDI\_OFF

and containing the character "Y".

An example of setting this environment variable in a C program follows:

```
setenv("_IDI_OFF","Y",1); /* disable IDIXCEE invocation*/
```

To reenable the invocation of Fault Analyzer through the IDIXCEE exit, you can set the \_IDI\_OFF environment variable to a value other than "Y", for example "N":

```
setenv("_IDI_OFF","N",1); /* re-enable IDIXCEE invocation */
```

For details of when the IDIXCEE exit is used, see “Exits for invoking Fault Analyzer” on page 223. The \_IDI\_OFF environment variable does not affect invocation of Fault Analyzer through any other exits.



---

## Chapter 31. Customizing Fault Analyzer by using user exits

In order to provide greater flexibility and control of Fault Analyzer operation, a set of user exit points have been created where user exits can get control during Fault Analyzer operation. The user exits can be written in REXX, Assembler or high-level languages. They are normally passed two data structures. The first is a common environment structure passed to all user exits which provides the general information fields for the fault currently being processed. The second structure is normally fields to the particular exit being called. Some of the fields are used to pass information to the exits and others are for the user exit to pass data or required actions back to Fault Analyzer. For exits written in REXX, the data is passed in stem variables rather than in structures since this approach is more manageable for REXX.

**Note:** REXX is the only supported programming language for Formatting user exits.

The exits can be used to perform functions such as dynamically selecting the history file data set or compile listing data sets. They can also be used to notify users by messages or e-mail that a fault has occurred plus many other uses.

Options settings and selections made in user exits affect the current analysis only.

The following user exits are available to users of Fault Analyzer:

- “Analysis Control user exit” on page 368
- “Analysis Control user exit (MVS SVC Dump registration)” on page 371
- “Compiler Listing Read user exit” on page 372
- “Message and Abend Code Explanation user exit” on page 377
- “Formatting user exit” on page 381
- “End Processing user exit” on page 393
- “End Processing user exit (Fault entry refresh)” on page 396
- “Notification user exit” on page 397
- “Notification user exit (MVS SVC Dump registration)” on page 402
- “IDIUTIL Import user exit” on page 403<sup>11</sup>
- “IDIUTIL Delete user exit” on page 404<sup>11</sup>
- “IDIUTIL ListHF user exit” on page 405<sup>11</sup>

Figure 144 on page 360 illustrates the exit points provided for real-time analysis, batch reanalysis, and interactive reanalysis, while Figure 145 on page 361 illustrates the exit points provided for the IDIUTIL batch utility.

---

11. Used with the IDIUTIL batch utility only.

## Customizing Fault Analyzer by using user exits

---

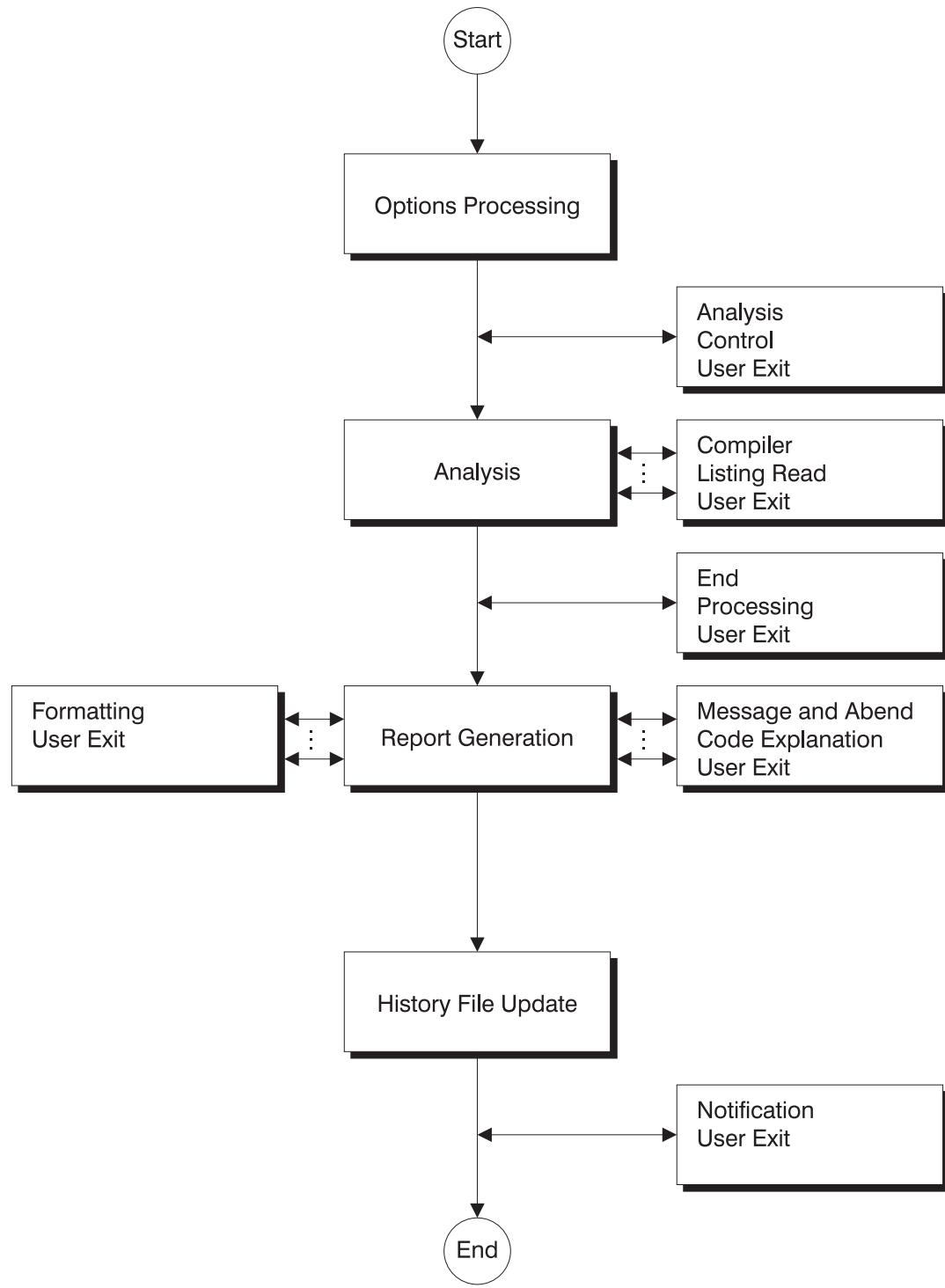


Figure 144. Fault Analyzer analysis exit points (IDIDA)

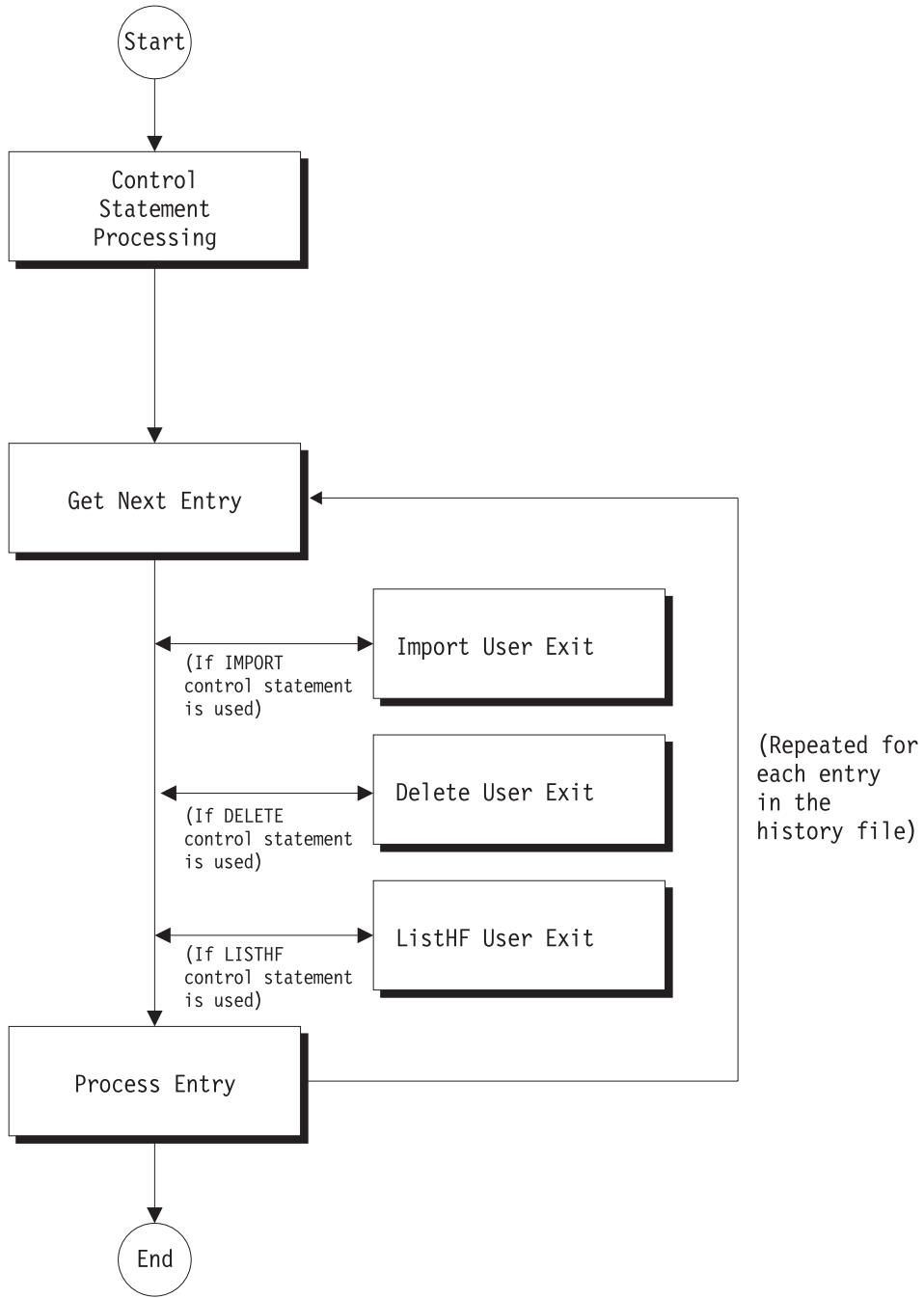


Figure 145. IDIUTIL batch utility exit points

User exits are specified to Fault Analyzer via the Exits option (see “Exits” on page 473), the DumpRegistrationExits option (see “DumpRegistrationExits” on page 466), or the RefreshExits option (see “RefreshExits” on page 496).

The IDIUTIL Import, IDIUTIL ListHF, and IDIUTIL Delete user exits operate with the IDIUTIL batch utility only. They use an Exits control statement—for details, see “EXITS control statement” on page 343.

## Invocation parameters

Two data areas are always passed to a user exit:

- A global environment data area (ENV).
- A data area specific to the type of user exit.

### Global environment data area (ENV)

The ENV data area provides information which is common to all exit types.

Two special fields are provided in this data area: USER\_1 and USER\_2. Initially, Fault Analyzer initializes these fields to blanks, but no reinitialization is performed between the invocation of the first and the last user exit. These fields can be used to pass information, for example the address of a data area, between exits.

A detailed description of the ENV data area is available in “ENV - Common exit environment information” on page 513.

### User exit type specific data area

For information about data areas that are specific to an exit type, refer to the individual exit types in “Descriptions of fault analysis user exit types” on page 367 or “Descriptions of IDIUTIL batch utility user exit types” on page 403.

---

## Supported exit programming languages

User exits can be written in REXX or in any language that permits a standard OS parameter list (R1 pointing to an array of fullwords) to be received on entry to the load module:

### REXX user exits

REXX user exits must be available via the IDIEXEC DDname. Data sets for this DDname can be provided in the DataSets option (see “DataSets” on page 457), or be specified through JCL DD statements in your job.

Modifiable parameter list values are always truncated to their defined field width on return from a user exit. Values that are shorter are delimited by adding a null character (X'00').

SYSPRINT type output from REXX EXECs, such as REXX messages and output from SAY or TRACE instructions, is suppressed by Fault Analyzer unless diagnostic tracing (see “Diagnostic tracing” on page 363 for details) is active. If you require messages to be written by REXX EXECs regardless of whether diagnostic tracing is active, use the IDIWTO command instead (see “IDIWTO command” on page 421 for details).

While developing new REXX exits, it is recommended to use IDITRACE in order to discover any problems that would otherwise not be externally visible.

Since a TSO/E REXX environment is not available to REXX user exits, the use of “Address TSO” commands is not supported. For information about what is supported, see *z/OS TSO/E REXX Reference*, chapter “Using REXX in different address spaces”, section “Writing execs that run in Non-TSO/E address spaces”.

### Load module exits

Load module exits must be available via the standard MVS search path

and cannot contain an LE main routine (C `main()` function or PL/I `PROC OPTIONS(MAIN)`) as they are executed under an existing LE enclave by the IDIDA subtask.

To assist with the writing of load module exits, Fault Analyzer provides the following parameter list data area mapping members in the samples data set:

Name	Language
<b>IDISXPLA</b>	Assembler
<b>IDISXPLC</b>	C
<b>IDISXPLB</b>	COBOL

**Note:** Due to COBOL language restrictions, all underscores ('\_') in parameter list field names have been substituted by dashes ('-').

<b>IDISXPLP</b>	PL/I
-----------------	------

Load module exits can be either reentrant or non-reentrant and should be link-edited RMODE(ANY). They are invoked in AMODE(31).

---

## Data area version checking

It is recommended that all user exits include a check for the expected version of any data areas used. This check is to ensure that incorrect processing does not result from changes to data areas that make them incompatible with earlier versions.

All user exit data areas include a VERSION field, whose current value at the time of writing the exit, can be obtained from the data area descriptions in Chapter 34, "Data areas," on page 505.

Examples of these checks are provided with all sample exits shown.

---

## Diagnostic tracing

To facilitate debugging information for user exits invoked via the Exits option, add the IDITRACE DDname to your JCL. For example:

```
//IDITRACE DD SYSOUT=*
```

(See "IDITRACE under CICS" on page 307 for an alternative method of activating this trace under CICS.)

## Tracing user exit parameter list values

When the IDITRACE DDname is allocated by the job step, the contents of all parameter lists are written to this DDname prior to the invocation of any user exit, and again upon return from the exit. Also, errors during validation of updated parameter list fields are identified by warning messages in the trace output.

Trace information is provided for all exit types for which exits are specified using the Exits option, and for which the execution mode of Fault Analyzer permits the

## Diagnostic tracing

exit type to be invoked. To facilitate trace information that shows the values of fields prior to exit invocation without first having to write the exit, use the special exit name 'NONE'.

The elapsed time in seconds for each exit invoked is provided.

An example of an exit trace containing a single Analysis Control user exit call to an exit named SAMPCTLX follows:

ANALYSIS CONTROL User Exit:

```
Parameter values prior to exit invocation:  
ENV.VERSION . . . . . : 0005  
EXIT_CALL_TYPE. . . . . : C  
FAULT_ID. . . . . :  
ABEND_DATE. . . . . : 2001/03/23  
ABEND_TIME. . . . . : 10:02:03  
DUP_DATE. . . . . :  
DUP_TIME. . . . . :  
ORIGINAL_DATE . . . . . :  
ORIGINAL_TIME . . . . . :  
REALTIME. . . . . : Y  
SYSTEM_NAME . . . . . : MVSA  
JOB_NAME. . . . . : CI03DA  
EXEC_PGM_NAME . . . . . : DFHSIP  
USER_ID . . . . . : CICSUSER  
GROUP_ID. . . . . : APC  
ABEND_CODE. . . . . : AEIL  
ABEND_REASON_CODE . . . . . :  
INVOCATION_ABEND_CODE . . . . . : AEIL  
ABEND_MODULE_NAME . . . . . : CICFRED  
CICS_TRANSACTION_ID . . . . . : FRED  
CICS_TASK_NUMBER. . . . . : 00026  
JOB_TYPE. . . . . : C  
JOB_CLASS . . . . . : A  
ACCOUNTING_FIELDS . . . . . :  
ACCOUNTING_INFO . . . . . :  
USER_1. . . . . :  
USER_2. . . . . :  
LOCK_FLAG . . . . . :  
LOOPPROTECTION_OPT. . . . . : Y  
WRITE_ROUTINE_EP. . . . . : X'00000000'  
INVOCATION_EXIT . . . . . : L  
STEP_NAME . . . . . : CICS  
JOB_ID. . . . . : JOB30073  
IMS_PROGRAM_NAME. . . . . :  
USER_NAME . . . . . :  
USER_TITLE. . . . . :  
APPLID. . . . . : QXPM2C53  
NETNAME . . . . . :  
TERMINAL. . . . . : SAMA  
TCB_ADDRESS . . . . . :  
CSA_ADDRESS . . . . . :  
TCA_ADDRESS . . . . . :  
IDTHIST . . . . . : FRED.DCAT  
CPU_HSECONDS. . . . . :  
CICS_VRM. . . . . :  
DB2_VRM . . . . . :  
IMS_VRM . . . . . :  
ZOS_VRM . . . . . :  
DUPLICATE_COUNT . . . . . : 00000  
POF_MODULE_NAME . . . . . :  
POF_MODULE_LKED_DATE. . . . . :  
POF_MODULE_LKED_TIME. . . . . :  
POF_CSECT_NAME. . . . . :  
POF_CSECT_OFFSET. . . . . : 00000000000  
POF_LOADED_FROM . . . . . :  
EXEC_LOADED_FROM. . . . . :  
MINIDUMP_PAGES. . . . . : 00000000000  
CTL.VERSION . . . . . : 0002
```

```

Exclude . . . . . : M
DETAIL_OPT. . . . . : N
DEFERREDREPORT_OPT. . . . . : AUTO
RETAINDUMP_OPT. . . . . : AUTO
SOURCE_OPT. . . . . : Y
IDIADATA_PRE. . . . . :
IDIADATA_JOB. . . . . :
IDIADATA_CFG. . . . . : FRED.SYSADATA
IDILC_PRE . . . . . :
IDILC_JOB . . . . . :
IDILC_CFG . . . . . : FRED.LISTING.C
IDILCOB_PRE . . . . . :
IDILCOB_JOB . . . . . :
IDILCOB_CFG . . . . . : FRED.LISTING.COOL
IDILCOBO_PRE . . . . . :
IDILCOBO_JOB . . . . . :
IDILCOBO_CFG. . . . . :
IDILANGX_PRE. . . . . :
IDILANGX_JOB. . . . . :
IDILANGX_CFG. . . . . : FRED.WDBLANGX
IDILPLI_PRE . . . . . :
IDILPLI_JOB . . . . . : USERA.JCLLIB
                                         USERA.LOAD
                                         USERA.TEXT
IDILPLI_CFG . . . . . : FRED.LISTING.PLI
IDILPLIE_PRE. . . . . :
IDILPLIE_JOB. . . . . :
IDILPLIE_CFG. . . . . :
LOCALE. . . . . : S370
FADATE. . . . . : N
IDISYSDB_PRE. . . . . :
IDISYSDB_JOB. . . . . :
IDISYSDB_CFG. . . . . :
Parameter values after return from exit SAMPCTLX (elapsed 0.03 seconds):
ENV.VERSION . . . . . : 0005
EXIT_CALL_TYPE. . . . . : C
FAULT_ID. . . . . :
ABEND_DATE. . . . . : 2001/03/23
ABEND_TIME. . . . . : 10:02:03
DUP_DATE. . . . . :
DUP_TIME. . . . . :
ORIGINAL_DATE . . . . . :
ORIGINAL_TIME . . . . . :
REALTIME. . . . . : Y
SYSTEM_NAME . . . . . : MVSA
JOB_NAME. . . . . : CI03DA
EXEC_PGM_NAME . . . . . : DFHSIP
USER_ID . . . . . : CICSUSER
GROUP_ID. . . . . : APC
ABEND_CODE. . . . . : AEIL
ABEND_REASON_CODE . . . . . :
INVOCATION_ABEND_CODE . . . . . : AEIL
ABEND_MODULE_NAME . . . . . : CICFRED
CICS_TRANSACTION_ID . . . . . : FRED
CICS_TASK_NUMBER. . . . . : 00026
JOB_TYPE. . . . . : C
JOB_CLASS . . . . . : A
ACCOUNTING_FIELDS . . . . . :
ACCOUNTING_INFO . . . . . :
USER_1. . . . . : ABCD
USER_2. . . . . : 0123
LOCK_FLAG . . . . . :
LOOPPROTECTION_OPT. . . . . : Y
WRITE_ROUTINE_EP. . . . . : X'00000000'
INVOCATION_EXIT . . . . . : L
STEP_NAME . . . . . : CICS
JOB_ID. . . . . : JOB30073
IMS_PROGRAM_NAME. . . . . :
USER_NAME . . . . . :
USER_TITLE. . . . . :
APPLID. . . . . : QXPM2C53

```

## Diagnostic tracing

```
NETNAME . . . . . : SAMA
TERMINAL . . . . . : SAMA
TCB_ADDRESS . . . . . :
CSA_ADDRESS . . . . . :
TCA_ADDRESS . . . . . :
IDIHIST . . . . . : USERA.DCAT
CPU_HSECONDS. . . . . :
CICS_VRM. . . . . :
DB2_VRM . . . . . :
IMS_VRM . . . . . :
ZOS_VRM . . . . . :
DUPLICATE_COUNT . . . . . : 00000
POF_MODULE_NAME . . . . . :
POF_MODULE_LKED_DATE. . . . . :
POF_MODULE_LKED_TIME. . . . . :
POF_CSECT_NAME. . . . . :
POF_CSECT_OFFSET. . . . . : 0000000000
POF_LOADED_FROM . . . . . :
EXEC_LOADED_FROM. . . . . :
MINIDUMP_PAGES. . . . . : 0000000000
CTL.VERSION . . . . . : 0002
Exclude . . . . . :
DETAIL_OPT. . . . . : S
DEFERREDREPORT_OPT. . . . . : N
RETAINDUMP_OPT. . . . . : AUTO
SOURCE_OPT. . . . . : Y
IDIADATA_PRE. . . . . :
IDIADATA_JOB. . . . . :
IDIADATA_CFG. . . . . : FRED.SYSADATA
IDILC_PRE . . . . . :
IDILC_JOB . . . . . :
IDILC_CFG . . . . . : FRED.LISTING.C
IDILCOB_PRE . . . . . :
IDILCOB_JOB . . . . . :
IDILCOB_CFG . . . . . : FRED.LISTING.COBOL
IDILCOBO_PRE. . . . . :
IDILCOBO_JOB. . . . . :
IDILCOBO_CFG. . . . . :
IDILANGX_PRE. . . . . :
IDILANGX_JOB. . . . . :
IDILANGX_CFG. . . . . : FRED.WDBLANGX
IDILPLI_PRE . . . . . :
IDILPLI_JOB . . . . . :
IDILPLI_CFG . . . . . : FRED.LISTING.PLI
IDILPLIE_PRE. . . . . :
IDILPLIE_JOB. . . . . : *** Data from 9945 byte buffer at address 17FF08F8 follows:
FRED.IDILPLIE.T001 . . . . . : FRED.IDILPLIE.T002
FRED.IDILPLIE.T003 . . . . . : FRED.IDILPLIE.T004
FRED.IDILPLIE.T005 . . . . . : FRED.IDILPLIE.T006
FRED.IDILPLIE.T007 . . . . . : FRED.IDILPLIE.T008
FRED.IDILPLIE.T009 . . . . . : FRED.IDILPLIE.T010
FRED.IDILPLIE.T011 . . . . . : FRED.IDILPLIE.T012
FRED.IDILPLIE.T013 . . . . . : FRED.IDILPLIE.T014
FRED.IDILPLIE.T015 . . . . . : FRED.IDILPLIE.T016
FRED.IDILPLIE.T017 . . . . . : FRED.IDILPLIE.T018
FRED.IDILPLIE.T019 . . . . . : FRED.IDILPLIE.T020
FRED.IDILPLIE.T021 . . . . . : FRED.IDILPLIE.T022
FRED.IDILPLIE.T023 . . . . . : FRED.IDILPLIE.T024
FRED.IDILPLIE.T025 . . . . . : FRED.IDILPLIE.T026
FRED.IDILPLIE.T027 . . . . . : FRED.IDILPLIE.T028
FRED.IDILPLIE.T029 . . . . . : FRED.IDILPLIE.T030
FRED.IDILPLIE.T031 . . . . . : FRED.IDILPLIE.T032
FRED.IDILPLIE.T033 . . . . . : FRED.IDILPLIE.T034
FRED.IDILPLIE.T035 . . . . . : FRED.IDILPLIE.T036
FRED.IDILPLIE.T037 . . . . . : FRED.IDILPLIE.T038
FRED.IDILPLIE.T039 . . . . . : FRED.IDILPLIE.T040
FRED.IDILPLIE.T041 . . . . . : FRED.IDILPLIE.T042
FRED.IDILPLIE.T043 . . . . . : FRED.IDILPLIE.T044
FRED.IDILPLIE.T045 . . . . . : FRED.IDILPLIE.T046
FRED.IDILPLIE.T047 . . . . . : FRED.IDILPLIE.T048
```

```
FRED.IDILPLIE.T049
FRED.IDILPLIE.T051
FRED.IDILPLIE.T053
FRED.IDILPLIE.T055
FRED.IDILPLIE.T057
FRED.IDILPLIE.T059
FRED.IDILPLIE.T061
FRED.IDILPLIE.T063
FRED.IDILPLIE.T065
FRED.IDILPLIE.T067
FRED.IDILPLIE.T069
FRED.IDILPLIE.T071
FRED.IDILPLIE.T073
FRED.IDILPLIE.T075
FRED.IDILPLIE.T077
FRED.IDILPLIE.T079
FRED.IDILPLIE.T081
FRED.IDILPLIE.T083
FRED.IDILPLIE.T085
FRED.IDILPLIE.T087
FRED.IDILPLIE.T089
FRED.IDILPLIE.T091
FRED.IDILPLIE.T095
FRED.IDILPLIE.T097
FRED.IDILPLIE.T099
FRED.IDILPLIE.T101
FRED.IDILPLIE.T103
FRED.IDILPLIE.T105
FRED.IDILPLIE.T107
FRED.IDILPLIE.T109
FRED.IDILPLIE.T111
FRED.IDILPLIE.T113
FRED.IDILPLIE.T115
FRED.IDILPLIE.T117
FRED.IDILPLIE.T119
FRED.IDILPLIE.T121

IDILPLIE_CFG. . . . . : FRED.IDILPLIE.T050
LOCALE. . . . . : S370
FADATE. . . . . : N
IDISYSDB_PRE. . . . . :
IDISYSDB_JOB. . . . . :
IDISYSDB_CFG. . . . . :
```

# Tracing REXX EXECs

If a user exit is written as a REXX EXEC, information written by the exit using SAY or TRACE instructions is merged with the Fault Analyzer exit parameter list trace records in the IDITRACE data set.

## **Descriptions of fault analysis user exit types**

The following provides descriptions of each user exit type available for use with Fault Analyzer in real time or fault reanalysis mode of execution. Table 10 shows which exit types that are applicable to each mode of execution.

*Table 10. Fault analysis user exit types applicable to execution modes*

User exit type	Real-time		Reanalysis		
	Normal	Dump registration	Batch		Interactive
			Normal	Refresh	
Analysis Control	Yes (1)	Yes (2)	Yes (1)	Yes (1)	Yes (1)
Compiler Listing Read	Yes (1)	No	Yes (1)	Yes (1)	Yes (1)
Message and Abend Code Explanation	Yes (1)	No	Yes (1)	Yes (1)	Yes (1)

## Descriptions of fault analysis user exit types

Table 10. Fault analysis user exit types applicable to execution modes (continued)

User exit type	Real-time		Reanalysis		Interactive
	Normal	Dump registration	Batch	Normal	
Formatting	Yes (1)	No	Yes (1)	Yes (1)	Yes (1)
End Processing	Yes (1)	No	No	Yes (3)	No
Notification	Yes (1)	Yes (2)	No	No	No

Notes:

- (1) Specified via the Exits option
- (2) Specified via the DumpRegistrationExits option
- (3) Specified via the RefreshExits option

## Analysis Control user exit

The following describes the Analysis Control user exit.

### Purpose

This exit can be used to examine and override current settings of the following options:

#### DataSets

Data sets for the following DDnames are provided:

- IDIADATA
- IDIHIST
- IDILANGX
- IDILC
- IDILCOB
- IDILCOBO
- IDILPLI
- IDILPLIE
- IDISYSDB

For the IDIADATA, IDILANGX, IDILC, IDILCOB, IDILCOBO, IDILPLI, IDILPLIE, and IDISYSDB DDnames, current data sets are provided as:

#### Pre-allocated data sets

These are data sets that were allocated prior to invoking Fault Analyzer, for example, via JCL DD statements. Data sets for each DDname is provided in the CTL.ddname\_PRE field.

The pre-allocated data set name fields are read-only and any changes are ignored by Fault Analyzer.

#### Job-allocated data sets

These are DataSets option specifications from the user options file (IDIOPTS). Data sets for each DDname is provided in the CTL.ddname\_JOB field.

Changes to the job-allocated data set name fields are honored by Fault Analyzer when it allocates this list of data sets.

#### Configuration data sets

These are DataSets option specifications from the IDICNF00 configuration member. Data sets for each DDname is provided in the CTL.ddname\_CFG field.

Changes to the configuration data set name fields are honored by Fault Analyzer.

The final concatenation order of data sets is:

1. Pre-allocated data sets
2. Job-allocated data sets
3. Configuration data sets

For the IDIHIST DDname, the current history file is provided in the ENV.IDIHIST data area field. The user exit can choose to change this data set name, in which case the supplied data set name is used as the history file for the current fault.

The same rules for the use of substitution symbols in data set names which apply to the DataSets option (see "Data set name substitution symbols" on page 460), also apply to data set names returned by an Analysis Control user exit.

#### **DeferredReport**

The status of the DeferredReport option in effect is identified as either 'Y' (DeferredReport is in effect) or 'N' (DeferredReport is not in effect) in the CTL.DEFERREDREPORT\_OPT data area field. Valid changes to this field override the current option setting.

Only applicable to real-time processing.

**Detail** The abbreviated form of the Detail option in effect is provided in the CTL.DETAIL\_OPT data area field. Valid changes to this field override the current option setting.

Not applicable to interactive reanalysis.

#### **Exclude**

The last matching Exclude criterion is provided in the CTL.EXCLUDE\_CRITERION data area field. If the fault is excluded from analysis based on a matching Exclude criterion, then the CTL.EXCLUDE data area field is initialized to 'Y'. This field can be modified by the exit.

Only applicable to real-time processing.

#### **Include**

The last matching Include criterion is provided in the CTL.INCLUDE\_CRITERION data area field. A blank Include criterion signifies the implicit product default, which is to include everything.

Only applicable to real-time processing.

**Locale** The current locale name is provided in the CTL.LOCALE data area field. Valid changes to this field override the current option setting.

The LOCALE option suboption, FADATE, is provided in the CTL.FADATE data area field. Valid changes to this field override the current option setting.

#### **RetainDump**

The value of the RetainDump option in effect is provided in the CTL.RETAINDUMP\_OPT data area field (the value provided in this field is the actual suboption of the RetainDump option, that is, "AUTO" or "ALL"). Valid changes to this field override the current option setting.

Note that an End Processing user exit (see "End Processing user exit" on page 393) might later override this option.

## Analysis Control user exit

Only applicable to real-time processing.

### Source

The status of the Source option in effect is identified as either 'Y' (Source is in effect) or 'N' (Source is not in effect) in the CTL.SOURCE\_OPT data area field. Valid changes to this field override the current option setting.

Only applicable to real-time processing.

In addition, the Analysis Control user exit can perform allocations of other data sets that might not be specified via options or provided in the passed data areas. In real time, one such allocation could be for IDIREPRT, which would allow an installation to control report destination attributes, such as the SYSOUT class. For more information on this type of IDIREPRT allocation, see "Combining Fault Analyzer real-time reports" on page 19, "Controlling the SYSOUT class of real-time reports" on page 19, and "Suppressing real-time reports" on page 19.

### When invoked

This exit is invoked after options processing has completed, and before the commencement of fault analysis.

### Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Analysis Control user exit.

**REXX:** Two stems are available to the exit:

- ENV.

Contains defined symbols for all fields in the ENV data area (see "ENV - Common exit environment information" on page 513).

- CTL.

Contains defined symbols for all fields in the CTL data area (see "CTL - Analysis Control user exit parameter list" on page 506).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

Address of an ENV data area (see "ENV - Common exit environment information" on page 513).

- 31-bit CTL address in word 2.

Address of a CTL data area (see "CTL - Analysis Control user exit parameter list" on page 506).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

## Example

The following is an example of an Analysis Control user exit that is written in REXX.

---

```
/* REXX */
/* Check data areas used */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if CTL.VERSION <> 2 then
  say 'Note: CTL data area version change - field usage review required!'
if ENV.REALTIME = 'Y' then do /* Exclude all MVSA jobs from analysis */
  if ENV.SYSTEM_NAME = 'MVSA' then
    CTL.Exclude = 'Y'
  /* Select a separate history file for DB2, IMS, and other jobs
   based on jobname */
  if SUBSTR(ENV.JOB_NAME,1,3) = 'DB2' then
    ENV.IDIHIST = 'MY.DB2.HIST'
  else if SUBSTR(ENV.JOB_NAME,1,3) = 'IMS' then
    ENV.IDIHIST = 'MY.IMS.HIST'
  else
    ENV.IDIHIST = 'MY.OTHER.HIST'
end
exit 0
```

---

*Figure 146. Sample REXX Analysis Control user exit*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(CONTROL(REXX(ABC)))
```

## Analysis Control user exit (MVS SVC Dump registration)

The following describes the dump registration Analysis Control user exit.

### Purpose

This exit can be used to examine and override current settings of the following options relevant to dump registration processing:

#### DataSets

Only the history file data set name is relevant to this exit. The current history file data set name is provided in the ENV.IDIHIST data area field. The user exit can choose to change this data set name, in which case the supplied data set name is used as the history file for the current fault. If the history file was pre-allocated, it is freed.

#### Exclude

The last matching Exclude criterion is provided in the CTL.EXCLUDE\_CRITERION data area field. If the fault is excluded from analysis based on a matching Exclude criterion, then the CTL.EXCLUDE data area field is initialized to 'Y'. This field can be modified by the exit.

#### Include

The last matching Include criterion is provided in the CTL.INCLUDE\_CRITERION data area field. A blank Include criterion signifies the implicit product default, which is to include everything.

## Analysis Control user exit (MVS SVC Dump registration)

### When invoked

This exit is invoked after options processing has completed, and before the writing of the MVS SVC dump registration fault entry.

### Parameters

See "Parameters" on page 370.

### Example

The following is an example of an Analysis Control dump registration user exit that is written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if CTL.VERSION <> 2 then
  say 'Note: CTL data area version change - field usage review required!'
/* Exclude all MVSA jobs from analysis */
if ENV.SYSTEM_NAME = 'MVSA' then
  CTL.Exclude = 'Y'
/* Select a separate history file for DB2, IMS, and other jobs
   based on jobname */
if SUBSTR(ENV.JOB_NAME,1,3) = 'DB2' then
  ENV.IDIHIST = 'MY.DB2.HIST'
else if SUBSTR(ENV.JOB_NAME,1,3) = 'IMS' then
  ENV.IDIHIST = 'MY.IMS.HIST'
else
  ENV.IDIHIST = 'MY.OTHER.HIST'
exit 0
```

---

Figure 147. Sample REXX Analysis Control dump registration user exit

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or an IDIOPTS user options file that is allocated to the IDIS subsystem, would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
DumpRegistrationExits(CONTROL(REXX(ABC)))
```

The DumpRegistrationExits option must be specified in the IDICNFxx parmlib member, or via an IDIOPTS DD statement in the IDIS subsystem JCL. The DumpRegistrationExits option is ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS region or batch job.

## Compiler Listing Read user exit

The following describes the Compiler Listing Read user exit.

### Purpose

This exit can be used to obtain source code information from sources other than compiler listings or Fault Analyzer side files stored as members in the usual IDI data sets (IDILANGX, IDILCOB, IDILPLI, and so on). For example, compiler listings that are stored in a compressed format, or that are accessible only via a proprietary access method.

Whenever source code information is required by Fault Analyzer, the steps shown in "Locating compiler listings or side files" on page 290 are performed. When the Compiler Listing Read user exit is shown invoked, this implies all specified and available exits. That is, all exits are first invoked to provide a side file and later, if

necessary, all exits are invoked to provide a compiler listing. Only the last side file or compiler listing provided by any exit is used.

Fault Analyzer provides the following information to the Compiler Listing Read user exit about the required source code information:

- The load module name. This name is available in the LST MODULE\_NAME data area field.<sup>12</sup>
- The load module data set name in the LST LOAD\_MODULE\_DSN data area field.
- The CSECT name. This name is available in the LST CSECT\_NAME data area field.
- The entry-point name. This name is available in the LST EP\_NAME data area field, truncated to a maximum of 256 characters. If truncated, the first 254 characters of the entry-point name is followed by a tilde (~) and the last character of the untruncated entry-point name.
- The compile date. This date is available in the LST COMPILE\_DATE data area field, in the format YYYY/MM/DD.
- The compile time. This time is available in the LST COMPILE\_TIME data area field, in the format HH:MM:SS.
- The listing type. This type is available in the LST LISTING\_TYPE data area field, as one of the following:

**L** Compiler listing or assembler SYSADATA file

**S** Fault Analyzer side file

The source code information passed back to Fault Analyzer must be of the type specified in this field.

- The language type. This type is available in the LST LANGUAGE\_TYPE data area field, as one of the following:
  - Assembler
  - C/C++
  - COBOL
  - OS/VS COBOL
  - PL/I
- The expected record format of listings or side files provided by the exit. This format is available in the LST RECFM data area field, as V, VB, VBA, F, FB, FBA, and so on.
- The expected logical record length of listings or side files provided by the exit. This length is available in the LST LRECL data area field, in decimal character format.

Based on the above information, a user exit can provide the requested listing or side file. This provision is done in one of the following ways:

- By passing one record at a time to Fault Analyzer via the LST data area:
  - Unless the name of a variable containing the listing record is passed on the IDIWRITE command in a REXX EXEC, the listing or side file data record must be provided in the DATA\_BUFFER field.
  - For variable-length records, the length of the record in DATA\_BUFFER must be provided in the DATA\_LENGTH field in decimal character format. This

---

12. See "Parameters" on page 375 for references to the LST data area.

## Compiler Listing Read user exit

length is not inclusive of any variable-length record descriptor word and must be less than or equal to the value in the LRECL field minus 4 bytes.

- For fixed-length records, the length of the record in DATA\_BUFFER is expected to match the LRECL field. Any value provided in the DATA\_LENGTH field is ignored.

or

- By placing the name of a sequential data set or PDS(E) with member specification in the DATA\_BUFFER field and setting DATA\_BUFFER\_DSN to Y. The data set name must be uppercased, must start at offset zero into the DATA\_BUFFER field, must be fully qualified, and must not contain surrounding quotes. Examples of valid data set names are:

MY.SEQ.DS  
MY.PDS.DS(MBR)

No characters other than blanks must follow the data set name (the DATA\_BUFFER field is initialized to blanks before the user exit is invoked).

If not providing a data set name, but instead passing back individual data records, then, having updated the LST data area with the data record information, two different methods are available for passing the data to Fault Analyzer depending on the exit type:

**REXX** The Fault Analyzer environment IDIWRITE command is used. To use this command, code a REXX statement as follows:

```
ADDRESS FAULTA 'IDIWRITE [var-name]'
```

Successful completion of the IDIWRITE command is indicated by a zero return code.

For detailed information about the IDIWRITE command, see “IDIWRITE command” on page 420.

### Load module

The address of a write routine is available in the ENV.WRITE\_ROUTINE\_EP data area field, as a hexadecimal 31-bit address.

This routine must be invoked with R1 pointing to a fullword containing the address of the ENV data area. For example, the following code fragments of user exits in different programming languages could be used to invoke the write routine:

#### Assembler:

```
ASMEXIT CSECT
...
L      R2,0(,R1)
USING ENV,R2
L      R3,4(,R1)
USING LST,R3
...
L      R15,ENV_WRITE_ROUTINE_EP
LA    R1,ENV_VERSION
ST    R1,**8
BAL   R1,**8
DC    F'0'
BALR  R14,R15
...
COPY  IDISXPLA
...
```

### C:

```
#include "SAMPLES(IDISXPLC)"
typedef void WRTN(ENV *pENV);
#pragma linkage(WRTN,OS)
int cexit(ENV *pENV, LST *pLST) {
    ...
    WRTN *write_rtn;
    write_rtn = (WRTN *)pENV->WRITE_ROUTINE_EP;
    write_rtn(pENV);
    ...
}
```

**COBOL:**

```
...
PROGRAM-ID. COBEXIT
...
LINKAGE SECTION.
    COPY IDISXPLB IN LIB.
    PROCEDURE DIVISION USING ENV, LST.
MAIN SECTION.
    ...
    CALL WRITE-ROUTINE-EP USING ENV.
    ...
END PROGRAM COBEXIT.
```

**PL/I:**

```
PLIEXIT: PROC (ENVPTR,LSTPTR) OPTIONS(BYVALUE,FETCHABLE) ;
Dcl (Envptr,Lstptr)      Pointer ;
%include syslib(IDISXPLP) ;
Dcl IDIWRITE            Entry Variable Options(Asm Byaddr) ;
...
Entryaddr(IDIWRITE) = Envptr->Write_Routine_EP ;
Call IDIWRITE (Envptr->Env) ;
...
End PLIEXIT ;
```

Return codes from the write routine are the same as the return codes from the IDIWRITE REXX command, except that RC=8 (syntax error) cannot be returned—see “IDIWRITE command” on page 420.

An indicator that can be used to cancel the usage of a listing or side file being provided by the user exit is available in the LST data area field DISREGARD\_EXIT\_LISTING. If this field is set to 'Y', any data records that might have been passed back to Fault Analyzer from the user exit is discarded. By means of this indicator, the user exit can prevent Fault Analyzer from using a partial listing in case errors occur while providing data records.

**When invoked**

This exit is invoked whenever source code information is required in any Fault Analyzer execution mode.

**Parameters**

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Compiler Listing Read user exit.

**REXX:** Two stems are available to the exit:

- ENV.

## Compiler Listing Read user exit

Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).

- LST.

Contains defined symbols for all fields in the LST data area (see “LST - Compiler Listing Read user exit parameter list” on page 522).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

Address of an ENV data area (see “ENV - Common exit environment information” on page 513).

- 31-bit LST address in word 2.

Address of an LST data area (see “LST - Compiler Listing Read user exit parameter list” on page 522).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### Example

The following is an example of a Compiler Listing Read user exit that is written in REXX. Given the Fault Analyzer request for a compiler listing file, it checks if a member name equal to the module name exists in a partitioned data set (myid.LISTING.TERSE) containing compressed compiler listings created using TRSMAIN.<sup>13</sup> If a member is found, it is de-compressed into a work data set and passed on to Fault Analyzer using the IDIWRITE command.

---

13. TRSMAIN is a free IBM compression utility—refer to the web site [ftp://service.boulder.ibm.com/s390/mvs/tools/packlib/](http://service.boulder.ibm.com/s390/mvs/tools/packlib/) README.HTML for more information.

---

```

/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if LST.VERSION <> 1 then
  say 'Note: LST data area version change - field usage review required!'
if LST.LISTING_TYPE = 'L' then do
  modnm = strip(LST MODULE NAME)
  "IDIALLOC DD(INFILE) DSN(myid.LISTING.TERSE("||modnm||")) SHR"
  if rc = 0 then do
    reclm = strip(LST.RECFM)
    lrecl = strip(LST.LRECL,L,0)
    "IDIALLOC DD(OUTFILE) DSN(myid.TEMP) NEW CATALOG SPACE(1,1) ",
      "RECFM("||reclm||") LRECL("||lrecl||") UNIT(SYSALLDA)"
    address linkmvs "trsmain unpack"
    address mvs "execio * diskr outfile (finis"
    do while queued() <> 0
      parse pull rec
      LST.DATA_LENGTH = length(rec)
      LST.DATA_BUFFER = rec
      "IDIWRITE"
    end
    "IDIFREE DD(INFILE,OUTFILE)"
  end
end
exit 0

```

---

*Figure 148. Sample REXX Compiler Listing Read user exit*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```

DataSets(IDIEXEC(X.Y.Z))
Exits(LISTING(REXX(ABC)))

```

## Message and Abend Code Explanation user exit

The following describes the Message and Abend Code Explanation user exit.

### Purpose

This exit can be used to provide explanations of messages and abend codes for the analysis report. A Message and Abend Code Explanation user exit, if available, receives control after Fault Analyzer has determined the availability of either a message explanation or an abend code explanation and prior to presenting the explanation in the analysis report. If Fault Analyzer was able to find an explanation using the softcopy books or override data sets, then the XPL.EXPLANATION\_AVAILABLE data area field<sup>14</sup> is set to 'Y'. The Message and Abend Code Explanation user exit can choose to:

- Retain the explanation if one was found to be available by doing nothing
- Provide an explanation if none was found, or replace one that was found

If a message explanation is being formatted by Fault Analyzer, the following information is provided in the XPL data area:

- The actual text of the message that was issued.

---

14. See "Parameters" on page 380 for references to the XPL data area.

## Message and Abend Code Explanation user exit

This text is available in the fields MESSAGE\_TEXT1 through MESSAGE\_TEXT10. MESSAGE\_TEXT1 is initialized to contain the first or only line of the message, MESSAGE\_TEXT2 through MESSAGE\_TEXT10 are used if the message consists of multiple lines.

**Note:** If the actual text of the message issued is not available, only the message ID is placed in the MESSAGE\_TEXT1 field.

If an abend code explanation is being formatted by Fault Analyzer, the following information is provided in the XPL data area:

- The abend code.

This code is available in the ABEND\_CODE field. The contents of this field depends on the abend type—see below for details.

- The abend reason code.

This code is available in the ABEND\_REASON\_CODE field as an 8-character hexadecimal value.

- The abend module name.

This name is available in the ABEND\_MODULE\_NAME field.

- The abend type.

This type is available in the ABEND\_TYPE field which contains one of the following values:

**C** Indicating a CICS transaction abend.

The field ABEND\_CODE contains a 4-character CICS abend code.

**D** Indicating a CICS dump code.

The field ABEND\_CODE contains a 4-character CICS dump code.

**S** Indicating a system abend.

The field ABEND\_CODE contains a 3-character hexadecimal left-justified system abend code.

**U** Indicating a user abend.

The field ABEND\_CODE contains a 4-character decimal user abend code.

Based on the above information, a user exit can provide a missing explanation or a replacement of the one found by Fault Analyzer. This provision is done by passing one record of the explanation at a time to Fault Analyzer via the XPL data area.

Unless the name of a variable containing the explanation record is passed on the IDIWRITE command in a REXX EXEC, the explanation data record must be provided in the DATA\_BUFFER field.

Having updated the XPL data area with the data record information, two different methods are available for passing the data to Fault Analyzer depending on the exit type:

**REXX** The Fault Analyzer environment IDIWRITE command is used. To use this command, code a REXX statement as follows:

```
ADDRESS FAULTA 'IDIWRITE [var-name]'
```

Successful completion of the IDIWRITE command is indicated by a zero return code.

## Message and Abend Code Explanation user exit

For detailed information about the IDIWRITE command, see “IDIWRITE command” on page 420.

### Load module

The address of a write routine is available in the ENV.WRITE\_ROUTINE\_EP data area field, as a hexadecimal 31-bit address.

This routine must be invoked with R1 pointing to a fullword containing the address of the ENV data area. For example, the following code fragments of user exits in different programming languages could be used to invoke the write routine:

#### Assembler:

```
ASMEXIT CSECT
...
L    R2,0(,R1)
USING ENV,R2
L    R3,4(,R1)
USING XPL,R3
...
L    R15,ENV_WRITE_ROUTINE_EP
LA    R1,ENV_VERSION
ST    R1,*+8
BAL   R1,*+8
DC    F'0'
BALR  R14,R15
...
COPY  IDISXPLA
...
```

#### C:

```
#include "SAMPLES(IDISXPLC)"
typedef void WRTN(ENV *pENV);
#pragma linkage(WRTN,OS)
int cexit(ENV *pENV, XPL *pXPL) {
...
WRTN *write_rtn;
write_rtn = (WRTN *)pENV->WRITE_ROUTINE_EP;
write_rtn(pENV);
...
}
```

#### COBOL:

```
...
PROGRAM-ID. COBEXIT
...
LINKAGE SECTION.
COPY IDISXPLB IN LIB.
PROCEDURE DIVISION USING ENV, XPL.
MAIN SECTION.
...
CALL WRITE-ROUTINE-EP USING ENV.
...
END PROGRAM COBEXIT.
```

#### PL/I:

```
PLIEXIT: PROC (ENVPTR,XPLPTR) OPTIONS(BYVALUE,FETCHABLE) ;
Dcl (Envptr,Xplptr)      Pointer ;
%include syslib(IDISXPLP) ;
Dcl IDIWRITE            Entry Variable Options(Asm Byaddr) ;
...
```

## Message and Abend Code Explanation user exit

```
Entryaddr(IDIWRITE) = Envptr->Write_Routine_EP ;
Call IDIWRITE (Envptr->Env) ;
...
End PLIEXIT ;
```

Return codes from the write routine are the same as the return codes from the IDIWRITE REXX command, except that RC=8 (syntax error) cannot be returned—see “IDIWRITE command” on page 420.

The total number of characters that form the message or abend code explanation must not exceed *max-chars* in the following formula:

$$\text{max-chars} = 32752 - (\text{num-recs} * 2)$$

where *num-recs* is the total number of records.

An attempt to pass back a record that would cause the maximum number of characters to exceed *max-chars* results in RC=4 being returned by IDIWRITE (or the write routine used by load module exits) and the record being ignored.

No formatting of text is performed by Fault Analyzer. All records are presented in the analysis report exactly as they were provided by the Message and Abend Code Explanation user exit.

Records whose length exceed the formatting width at the time of presentation are wrapped at the current indentation, as determined by the leading number of blank characters in each record.

If more than one Message and Abend Code Explanation user exit provides a specific message or abend code explanation, then only the last explanation is used.

### When invoked

This exit is invoked during formatting of the analysis report, regardless of the execution mode of Fault Analyzer.

### Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Message and Abend Code Explanation user exit.

**REXX:** Two stems are available to the exit:

- ENV.

Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).

- XPL.

Contains defined symbols for all fields in the XPL data area (see “XPL - Message and Abend Code Explanation user exit parameter list” on page 530).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

Address of an ENV data area (see “ENV - Common exit environment information” on page 513).

- 31-bit XPL address in word 2.

Address of an XPL data area (see “XPL - Message and Abend Code Explanation user exit parameter list” on page 530).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### Example

The following is an example of a Message and Abend Code Explanation user exit that is written in REXX.

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if XPL.VERSION <> 1 then
  say 'Note: XPL data area version change - field usage review required!'
parse var XPL.MESSAGE_TEXT1 msgid msgtext
if msgid = 'MYMSG01' then do
  rec = 'This message indicates that:'
  'IDIWRITE rec'
  rec = ' - A serious problem has occurred'
  'IDIWRITE rec'
  rec = ' - Any data produced should be ignored'
  'IDIWRITE rec'
end
```

Figure 149. Sample REXX Message and Abend Code Explanation user exit

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(MSGXPL(REXX(ABC)))
```

## Formatting user exit

**Note:** The normal load module format user exit is not supported as a Formatting user exit. However, a special load module format user exit, IDIXUFMT, is available. For details, see “The IDIXUFMT load module Formatting user exit” on page 435.

### Purpose

This exit can be used to create a user-specific section in the analysis report, which can, for example, be used to format data areas which are specific to the application environment being analyzed.

These are different ways to invoke exits of this type:

1. By using the Exits option.

This option results in a User analysis report section being inserted between the System-Wide Information section and the Abend Job Information section. For details on specifying the Exits option, see “Exits” on page 473.

## Formatting user exit

In the real-time or batch reanalysis report, the user analysis section is inserted between the System-Wide Information section and the Abend Job Information section as shown in the following:

```
⋮  
<H1> S Y S T E M - W I D E   I N F O R M A T I O N  
⋮  
<H1> U S E R  
⋮  
<H1> I B M   F A U L T   A N A L Y Z E R   A B E N D   J O B   I N F O  
⋮
```

In the interactive reanalysis report, the user analysis section is selectable from the Interactive Reanalysis Report display as shown in the following:

1. Synopsis
2. Event Summary
3. System-Wide Information
4. **User**
5. Abend Job Information
6. Options in Effect

The data from all invoked Formatting user exits is included in the report.

The default "User" heading can be changed by setting UFM.USEROPTIONTITLE to a different value. The last value set by any Formatting user exit is used.

### 2. By using the EXEC command.

This option is only available from the interactive reanalysis report. For details on using the EXEC command, see "EXEC" on page 66.

### 3. By making available a load module in an APF-authorized library named IDIXUFMT. For more information about this type of Formatting user exit, see "The IDIXUFMT load module Formatting user exit" on page 435.

The Formatting exit is initially provided with information about the point-of-failure event in the exit-specific UFM data area. (See "Parameters" on page 383 for references to the UFM data area.) However, information for any event can be obtained by using the IDIEventInfo command. This process is described in "IDIEventInfo command" on page 416.

Information about the existence of a load module, including its load address and length, can be obtained using the IDIModQry command. This process is described in "IDIModQry command" on page 418.

To obtain storage from the analyzed environment, or to write data to the report, extra commands are available:

#### Evaluate

This command is used to retrieve storage from the analyzed environment.

Details of this command are provided in "Evaluate command" on page 409.

#### List

This command is used to print storage in the report.

Details of this command are provided in "List command" on page 422.

#### Note

This command is used to print a line of text in the report.

Details of this command are provided in "Note command" on page 423.

In addition to using the List and Note commands, a HTML-like tag language is available for greater flexibility in formatting the information for the report. Details of these tags are provided in “Formatting tags” on page 425.

The tagged text is passed back to Fault Analyzer using the IDIWrite command, in one of three different ways:

1. Using a quoted string

Example:

```
IDIWrite '<p>Paragraph text.'
```

Either single quotes (') or double quotes ("") can be used to enclose the string. However, both characters must be of the same type.

If the string contains characters of the same type as those used to enclose the string, then these must be repeated twice. That is, to pass back the string

```
'The TCB's address is not zero'
```

specify

```
'The TCB''s address is not zero'
```

2. Using a variable

Example:

```
data = '<p>Paragraph text.'
IDIWrite data
```

3. Using the UFM data area

Example:

```
UFM.DATA_BUFFER = '<p>Paragraph text.'
UFM.DATA_LENGTH = length(UMF.DATA_BUFFER)
IDIWrite
```

This method is primarily provided for non-REXX exits.

The List and Note commands can be used intermixed with the tag language without any formatting side effects.

## When invoked

This exit is invoked during formatting of the analysis report, regardless of the execution mode of Fault Analyzer. Additionally, exits of this type can be invoked on demand from the interactive reanalysis report by using the EXEC command—for details, see “User-specific report formatting” on page 164.

## Parameters

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Message and Abend Code Explanation user exit.

Two stems are available to the exit:

- ENV.

Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).

- UFM.

Contains defined symbols for all fields in the UFM data area (see “UFM - Formatting user exit parameter list” on page 525).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

## Formatting user exit

### Example 1

The following is an example of a Formatting user exit that is written in REXX. The exit is provided in softcopy format as member IDISUFM1 in data set IDI.SIDISAM1.

```
/*-----*/
/* Sample Fault Analyzer Formatting user exit to display TCB information. */
/*
/* This sample EXEC does the following:
/*
/* 1) Takes the current TCB address, passed in the ENV structure, and
/*     from this determines the Jobstep TCB address.
/* 2) It then follows the daughter and sibling TCB chains for all TCBs
/*     extracting each TCB CPU time (TCBTTIME), program name and completion
/*     code.
/*
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UFM.VERSION <> 1 then
  say 'Note: UFM data area version change - field usage review required!'
"IDIWRITE '<P>Fault Analyzer Formatting User Exit Example</P>'"  
Numeric Digits 30
ThisTCB = ENV.TCB_ADDRESS
"IDIWRITE '<th>Display TCB Information</th>'"
"IDIWRITE '<L> </L>'"
TCBCount = 0
Call GetTCBList           /* Get list of all TCBs under JOBSTEP TCB      */
If TCBCount = 0 Then      /* No TCB could be found                         */
  "IDIWRITE '<L>Unable to determine TCBs in dump</L>'"  
Else
  "IDIWRITE '<P>Total number of TCBs found = " TCBCount "</P>'"  
Exit
/*
/* GetTCBList: Take the TCB address in variable ThisTCB and from this
/*             get the JobStep TCB address from offset 7C. Then process
/*             all daughter and sibling TCB chains.                         */
/*
GetTCBList:
temp = d2x(x2d(ThisTCB)+x2d('7C'))
"EVALUATE address(\"temp\") length(4) REXX(STORAGE(JSTCB))"  
"IDIWRITE '<L>Jobstep TCB = <ADDR" JSTCB "></ADDR></L>'"  
"IDIWRITE '<L> </L>'"

Call ProcessDaughters(JSTCB 1)

Return

/*
/* ProcessDaughters:
/*
/*
ProcessDaughters:Procedure Expose TCBCount
Arg TCB Indent
stop = 0

CPUTime = GetTime(TCB)      /* Extract TCBTTIME for this TCB          */
Pgm = GetPgm(TCB)          /* Try to get program name for current RB */
CompCode = GetCompCode(TCB) /* If there is a RTM2 get TCB completion code*/

Info = Left(Pgm,8) CPUTime CompCode

TCBCount = TCBCount + 1
TC = Right(TCBCount,3,'0')
EC = d2x(x2d(TCB)+x2d('100'))/* Validate TCB Eyecatcher offset          */
"EVALUATE address(\"EC\") length(4) REXX(STORAGE(X) UNFORMATTED)"  
If x <> 'TCB' Then
  Do
```

```

"IDIWRITE '<L>Invalid TCB at Address <ADDR "TCB"></ADDR></L>'"
"IDIWRITE '<L>Eyecatcher" x "</L>'"
Return
End

"IDIWRITE '<L>" TC Copies(' ',Indent) "<ADDR "TCB "></ADDR>" Info "</l>'"
Indent = Indent + 4

/* Having printed the current TCB see if it */
/* has a Daughter TCB and if so process its */
/* siblings and its daughters */
/*
"EVALUATE ADDRESS("TCB") length(4) REXX(STORAGE(DaughterTCB)) POSITION(x'88')"
If DaughterTCB <> '00000000' Then
Do
Call ProcessSiblings(DaughterTCB Indent)
Call ProcessDaughters(DaughterTCB Indent)
End
Return

/*
/* ProcessSiblings: Check if the TCB has any siblings and if so process */
/* those siblings plus any daughters of those siblings */
*/
ProcessSiblings:Procedure Expose TCBCount
Arg TCB Indent

"EVALUATE ADDRESS("TCB") length(4) REXX(STORAGE(SiblingTCB)) POSITION(x'80')"
If SiblingTCB <> '00000000' Then
Do
Call ProcessSiblings(SiblingTCB Indent)
Call ProcessDaughters(SiblingTCB Indent)
End
Return

/*
/* GetTime: For the given TCB extract TCBTTIME and convert to seconds. */
*/
GetTime:Procedure
Arg TCBAddress
TCBTTIME = d2x(x2d(TCBAddress)+x2d('13C'))
"EVALUATE address("TCBTTIME") length(8) REXX(STORAGE(X))"
Seconds = x2d(x) / 4096 / 1048576
Seconds = ((Seconds * 1000) % 1) / 1000
TheTime = '<HP>' || Seconds || '</HP>'
Return(TheTime)

/*
/*GetPgm: For a given TCB scan down the RB chain looking for the last */
/* one i.e. next RB pointer = TCB address. If the last RB is a */
/* PRB then get the CDE address and from that extract the program */
/* name. */
*/
GetPgm:Procedure
Arg TCBAddress
FoundLastRB = 0
RB = d2x(x2d(TCBAddress)+x2d('0'))
"EVALUATE address("RB") length(4) REXX(STORAGE(FAVar))"
CurrentRB = FAVar           /* Get the current RB */
RBCount = 0

Do Until FoundLastRB      /* Loop through RB blocks looking for last */
  "EVALUATE address("CurrentRB") length(4) REXX(STORAGE(FAVar)) POSITION(X'1C')"
  FAVar = '00' || Substr(FAVar,3)      /* clear top byte */
  If FAVar = TCBAddress Then
    FoundLastRB = 1
  Else

```

## Formatting user exit

```
Do
    RBCount = RBCount + 1
    CurrentRB = FAVar
End
If RBCount > 100 Then      /* Keep count just in case we have gone adrift */
Do
    Pgm = 'RB count exceeded'
    Return(Pgm)
End
End
                                         /* Having code the last RB is it a PRB      */
"EVALUATE address("CurrentRB") length(1) REXX(STORAGE(RBType)) POSITION(X'A')"
If RBType = '00' Then
Do
    "EVALUATE address("CurrentRB") length(4) REXX(STORAGE(CDE)) POSITION(X'C')"
    CDE = '00' || Substr(CDE,3)
    "EVALUATE address("CDE") length(8) REXX(STORAGE(PGM)) POSITION(X'8')"
    "CHARACTER"
    If pgm = '.....' Then
        pgm = 'Unknown'
    End
Else
Do
    pgm = 'Unknown'
End
Return(Pgm)

/*
/*-----*/
/* GetCompCode: For a given TCB check the RTM2 address and if non zero */
/*              extract the completion code from the TCB.                 */
/*-----*/
GetCompCode:Procedure
Arg TCBAddress
CompCode = ''
RTM2 = d2x(x2d(TCBAddress)+x2d('E0'))
"EVALUATE address("RTM2") length(4) REXX(STORAGE(FAVar))"
If FAVar <> '00000000' Then
Do
    "EVALUATE address("TCBAddress") length(4) REXX(STORAGE(FAVar))",
    "POSITION(X'10')"
    CompCode = FAVar
End
Return(CompCode)
```

If the above sample exit existed as member IDISUFM1 in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked during analysis:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(FORMAT(REXX(IDISUFM1)))
```

Alternatively, the exit could be invoked from the interactive reanalysis report by entering the EXEC IDISUFM1 command.

### Example 2

The following is another example of a Formatting user exit that is written in REXX. The exit is provided in softcopy format as member IDISUFM2 in data set IDI.SIDISAM1.

```
/*
/*-----*/
/* Sample Fault Analyzer Formatting user exit to display CICS CWA info. */
/*
/* This sample EXEC does the following:
/*
/* 1) Displays the TCB, CSA and TCA addresses in a definition list format */
/* 2) Extracts the CWA address from the CSA                           */
*/
```

```

/* 3) If running in realtime the CWA is displayed or if in re-analysis */
/*      a point-and-shoot field is created which when selected by the user */
/*      will show a Dump Storage display of the CWA address. */
/* 4) A DSECT map of the CWA is then displayed by specifying the name */
/*      of the DSECT and its source on a LIST command. */
/* 5) In this example the CWA contains a EIBDATE and EIBTIME which are */
/*      extracted and displayed in a more conventional format. */
/* 6) Also in this example the CWA contains the address of a storage */
/*      area which was EXEC CICS GETMAINed with the SHARED option. This */
/*      storage area would not normally be included in the Fault Analyzer */
/*      minidump so this exec issues an EVALUATE for the shared storage */
/*      address and hence ensuring the require storage area is included */
/*      in the minidump. */
/* 7) The shared storage area is then displayed or a point-and-shoot */
/*      field is created depending whether this EXEC is running in */
/*      realtime or interactive mode. */
/*-----*/
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UFM.VERSION <> 1 then
  say 'Note: UFM data area version change - field usage review required!'
"IDIWRITE '<P>Sample Formatting User Exit</P>'"
"IDIWRITE '<th>Common Addresses</th>'"

tcb = ENV.TCB_ADDRESS
csa = ENV.CSA_ADDRESS
tca = ENV.TCA_ADDRESS

"IDIWRITE '<DL>'"
"IDIWRITE '<DT>TCB address</DT><DD><addr "tcb"></addr></DD>'"
"IDIWRITE '<DT>CSA address</DT><DD><addr "csa"></addr></DD>'"
"IDIWRITE '<DT>TCA address</DT><DD><addr "tca"></addr></DD>'"
"IDIWRITE '</DL>'"
           /* The CICS CWA address is x'E4' into CSA */
cwa = d2x(x2d(csa)+x2d('E4'))
"IDIWRITE '<P>CWA address is hex E4 into CSA => <hp>"Right(cwa,8,'0')"</hp>'"
           /* Use EVALUATE to get the CWA address */
"EVALUATE address("cwa") length(4) REXX(STORAGE(FAVar))"
cwa = FAVar

"IDIWRITE '<AREA INDENT=5>'"
"IDIWRITE '<P>CWA address is <hp>"cwa"</hp></p>'"
           /* If this EXEC is being called in real */
           /* time then use the DUMP tag to display */
           /* the CWA contents. If interactive then */
           /* create a point-and-shoot field. */
if ENV.REALTIME = 'Y' Then
  "IDIWRITE '<P><DUMP" cwa " 512 >Dump of CWA</DUMP></P>'"
Else
  "IDIWRITE '<P><ADDR" cwa ">Press Enter Here to see CWA</ADDR></P>'"
"IDIWRITE '</AREA>'"

"IDIWRITE '<th>DSECT Map of CWA is as follows:</th>'"
dsect = 'CWA'          /* Name of DSECT to be mapped */
dsn = 'SIMCOCK.TEST.COPY(CWA)' /* Specify location of DSECT */

"LIST" cwa "DSECT(" || dsect dsn || ")"
           /* Now format the date and time values */
           /* from the CWA */
TheTime = d2x(x2d(cwa)+x2d('04'))
"EVALUATE address("TheTime") length(4) REXX(STORAGE(temp))"
FormattedTime = Substr(temp,2,2)':'Substr(temp,4,2)':'Substr(temp,6,2)
"IDIWRITE '<P>Formatted Time <hp>"FormattedTime"</hp></P>'"

TheDate = d2x(x2d(cwa)+x2d('00'))
"EVALUATE address("TheDate") length(4) REXX(STORAGE(temp))"

```

## Formatting user exit

```
FormattedDate = Substr(temp,3,5)
If FormattedDate <> '00000' Then
    NormalDate = Date('N',FormattedDate,'J')
Else
    NormalDate = FormattedDate
"IDIWRITE '<P>Formatted Date <hp>"NormalDate"</hp></P>'"

/* Ensure shared storage is in minidump */
Comstor = d2x(x2d(cwa)+x2d('58'))
"EVALUATE address("Comstor") length(4) REXX(STORAGE(FAVar))"           /* Ensure shared storage is in minidump */
/* by issuing an EVALUATE for the */                                     /* */
/* SHARED storage address. */                                         /* */

If ENV.REALTIME = 'Y' Then
    "EVALUATE address("FAVar") length(1024) REXX(STORAGE(temp))"          /* Now display the contents of the */
/* SHARED storage. */                                                 /* */

If ENV.REALTIME = 'Y' Then
    "IDIWRITE '<P><DUMP" FAVar " 512 >Dump of Shared Storage</DUMP></P>'"      /* */
Else
    "IDIWRITE '<P><ADDR" FAVar ">Press Here to see Shared Storage</ADDR></P>'"      /* */
Exit
```

If the above sample exit existed as member IDISUFM2 in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(FORMAT(REXX(IDISUFM2)))
```

Alternatively, the exit could be invoked from the interactive reanalysis report by entering the EXEC IDISUFM2 command.

## Example 3 (Hogan)

This is a sample Formatting user exit that is written in REXX. Use with Hogan applications. The exit is provided in softcopy format as member IDISUFM4 in data set IDI.SIDISAM1.

```
-----*/
/* Although the name of this sample is IDISUFM4 it may be more          */
/* convenient to rename it to a shorter, more relevant name, HOGAN for      */
/* example. */                                                               */
-----*/
Arg TCAAddress

trace 0

Call Initialise

"IDIWRITE '<DL> ''"
"IDIWRITE '<DT>Job Type  </DT><DD>"JT"</DD>'"
"IDIWRITE '<DT>TCB address</DT><DD><addr "tcb"></addr></DD>'"
"IDIWRITE '</DL>'

/* Firstly check to see which */                                         */
/* 'PEM' module is loaded. */                                              */
/* There should only be 1. */                                             */

Do i = 1 To PEMModule.0
    "IDIMODQRY NAME("PEMModule.i") LP(LPADDR) MODLEN(THELEN)"
    If rc = 0 Then
        Do
            PEMCount = PEMCount + 1
            PA.PEMCount = LPADDR
            PL.PEMCount = THELEN
            PN.PEMCount = PEMModule.i
```

```

        PEMMod = i
    End
End

Select
When PEMCount = 0 Then
Do
    "IDIWRITE '<P> None of the following modules were found</P>'"
    "IDIWRITE '<UL>'"
    Do i = 1 to PEMModule.0
        "IDIWRITE '<LI>"PEMModule.i "</LI>'"
    End
    "IDIWRITE '</UL>'"
End
When PEMCount > 1 Then
Do
    "IDIWRITE '<P>"PEMCount "PEM modules found - abort</P>'"
    Do i = 1 to PEMCount
        "IDIWRITE '<P>"PN.i "Load Point:"PA.i "Length:"PL.i "</P>'"
    End
End
Otherwise
Do
    PEMAddress = PA.1
    PEMLength = PL.1
    Call Discover_Hogan_Information
End
End
Exit

/*
/* Discover_Hogan_Information
/* Having determined that there is only one PEM module loaded we will now
/* proceed to determine the Hogan information.
*/
Discover_Hogan_Information:

"IDIWRITE '<P>"PEMModule.PEMMod "loaded at address" PEMAddress,
    "Length" PEMLength"</p>'"
UmbrellaVersion = Get_Hogan_Umbrella_Version()
If UmbrellaVersion = 0 Then
Do
    "IDIWRITE '<P> Could not determine Umbrella version</P>'"
    Exit
End

"IDIWRITE '<P> Umbrella version" UmbrellaVersion "<P>'"

Call Get_ITCB          /* Internal Transaction Control Block      */
Call Get_UTCB          /* User Transaction Control Block       */
Call Get_UPCB          /* User Program Control Block         */
Call Map_ITCB          /* */

Return

/*
/* Get_Hogan_Umbrella_Version
/* From the PEM load point scan for P49004 eyecatcher.
/* Format of eyecatcher P49004 UBM401
*/
Get_Hogan_Umbrella_Version:

BytesToScan = 512
Version = 0
HexEyeCatcher = C2X('P49004')
"EVALUATE ADDRESS("PEMAddress") LENGTH("BytesToScan") REXX(STORAGE(X))"
If rc = 0 Then

```

## Formatting user exit

```
Do
  Found = 0
  Do i = 1 to (Length(X) - 8) While Found = 0
    If Substr(x,i,Length(HexEyeCatcher)) = HexEyeCatcher Then
      Do
        Found = 1
        Version = Substr(x,(i+Length(HexEyeCatcher)+12),6)
        Version = X2C(Version)
      End
    End
  End
Return(Version)

/*-----*/
/* Initialise:
/*-----*/
Initialise:

PEMModule.0 = 7
PEMModule.1 = 'BATCHPEM'
PEMModule.2 = 'CICSPEM'
PEMModule.3 = 'BMPPEM'
PEMModule.4 = 'DLIPEM'
PEMModule.5 = 'IMSPEM'
PEMModule.6 = 'IMSPEMF'
PEMModule.7 = 'SYSB2PEM'

PEMMod = 0
PEMAddress = 0
PEMLength = 0
PEMCount = 0
PL. = 0
PA. = 0
PN. = ''

JobType. = 'Unknown'
type = 'B'
Jobtype.type = 'Batch Job'
type = 'S'
Jobtype.type = 'Started Task'
type = 'T'
Jobtype.type = 'TSO'
type = 'C'
Jobtype.type = 'CICS Transaction'
type = 'I'
Jobtype.type = 'CICS System'
type = 'D'
Jobtype.type = 'Dump Registration'

TWAOffset = 'EC'
TWALength = 'F0'

tcb = ENV.TCB_ADDRESS
csa = ENV.CSA_ADDRESS
tca = ENV.TCA_ADDRESS
job = ENV.JOB_TYPE
JT = Jobtype.job

If job = 'I' Then
  tca = TCAAddress

UTCBOffset. = '000'
UTCBOffset.3 = '8E8'
UTCBOffset.4 = 'AF8'

Return
```

```

/*
/* Get_ITCB
/* The ITCB is determined differently for CICS and non-CICS and for non-CICS */
/* depends on the Umbrella version.
*/
/*
Get_ITCB:

If job = 'C' | job = 'I' Then /* CICS system or transaction */
Do
    /* If this is a SDUMP then we need */
    /* to have been passed in the TCA */
    /* address of the abending (Hogan) */
    /* task. From this we can get the */
    /* TWA address which has the ITCB */
    /* address as the first word. */
If tca <> '' Then
Do
    text = "<P>TCA address<addr "tca"></addr></P>",
    "IDIWRITE text"

    OffsetTWAAddr = 'EC'
    OffsetTWALength = 'F0'
    FieldAddr = d2x(x2d(tca)+x2d(OffsetTWALength))
    "EVALUATE address("FieldAddr") length(4) REXX(STORAGE(X))"
    TWALength = x
    If TWALength > 0 Then
        Do
            FieldAddr = d2x(x2d(tca)+x2d(OffsetTWAAddr))
            "EVALUATE address("FieldAddr") length(4) REXX(STORAGE(X))"
            TWAAddress = x
            If TWAAddress > 0 Then
                Do
                    "EVALUATE address("TWAAddress"),
                     "length("TWALength"),
                     "REXX(STORAGE(X))"
                    ITCBAddr = Substr(x,1,8)
                    "IDIWRITE '<P>ITCB Address <addr " ITCBAddr,
                     "></addr></P>'"

                End
            End
        End
    End
Else
Do
    "IDIWRITE '<P>User TCA address must be provided for CICS SDUMP.</P>'"
End
End
Else
Do
    /* Not CICS system or transaction */
End
Do
End
Return

/*
/* Get_UTCB
/* The UTCB is contained in the ITCB. Starting at the following offsets:
/* Umbrella version      Offset
/*      V3                  0x8E8
/*      V4                  0xAF8
*/
Get_UTCB:
x = Left(version,1)
FieldOffset = UTCBOffset.x
FieldAddr = d2x(x2d(ITCBAddr)+x2d(FieldOffset))
"EVALUATE address("FieldAddr") length(2) REXX(STORAGE(XX))"

```

## Formatting user exit

```
eyecatcher = "First 2 bytes 0x"XX" (" x2c(xx) ")"
UTCBAddr = FieldAddr
"IDIWRITE '<P>UTCB Address <addr "UTCBAddr"></addr>' eyecatcher "</P>'"

Return

/*-----
/* Get_UPCB
/* The UPCB address is at offset x'64' in the ITCB
/*-----*/
Get_UPCB:
FieldOffset = '64'
FieldAddr = d2x(x2d(ITCBAddr)+x2d(FieldOffset))
"EVALUATE address("FieldAddr") length(4) REXX(STORAGE(X))"
UPCBAddr = x
"IDIWRITE '<P>UPCB Address <addr "UPCBAddr"></addr> </P>'"

Return
/*-----
/* Map_ITCB:
/* Use DSECT function to map ITCB. Note this function passes the name of the */
/* data set to use for the DSECT. This needs to changed according to local   */
/* naming conventions plus may need to be dynamically altered, as shown below,*/
/* according to the Umbrella version.
/*-----*/
Map_ITCB:
dsect = 'P49$ITCB'           /* Name of DSECT to be mapped          */
                               /* Specify location of DSECT          */
dsn = 'SIMCOCK.HOGANV' || Left(Version,1) || '.DSECTS(HOGAN)'

"LIST" ITCBAddr "DSECT(" || dsect dsn || ")"
Return
```

If the above sample exit existed as member HOGAN in data set X.Y.Z, then providing these options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(FORMAT(REXX(HOGAN)))
```

Alternatively, the exit could be invoked from the interactive reanalysis report by entering the EXEC HOGAN command.

## Example 4 (Batch MVS dump registration)

This is a sample Formatting user exit that is written in REXX. Use it with an MVS dump analysis batch job to create a fault entry in a designated history file. Optionally, the exit can write a WTO message or send an email with information about the fault entry that was created. The exit is provided in softcopy format as member IDISUFM5 in data set IDI.SIDISAM1.

```
emailaddr ='ibmuser@au1.ibm.com'           /* <== change */
histdsn  ='idi.hist'                      /* <== change */
smtpdest ='PRDSYS0.SMTP'                  /* <== change */
smtpdomain='au1.ibm.com'                  /* <== change */
if ENV.VERSION<4 then
  say 'Note: ENV data area version change - field usage review required!'
  'IDIRegisterFaultEntry IDIHIST('histdsn')'
if RC=0 then do
  /* Send notifications if fault entry successfully created */
  msg = 'Fault entry' ENVFAULT_ID,
        'created in history file' ENV.IDIHIST,
        'for job' ENVJOB_NAME,
        'abend' ENVABEND_CODE,
        'which occurred on' ENVABEND_DATE,
        'at' ENVABEND_TIME
```

```

/* Send WTO message */
'IDIWTO REGDUMP:' msg
/* Send email */
queue "HELO" smtpdomain
queue "MAIL FROM:<"emailaddr">"
queue "RCPT TO:<"emailaddr">"
queue "DATA"
queue "DATE: "date('N') time('C')"
queue "FROM:<"emailaddr">"
queue "TO:<"emailaddr">"
queue "Subject: REGDUMP"
queue msg
n=queued()
"IDIALLOC DD(EMAIL) SYSOUT(A) DEST("smtpdest")"
address mvs "EXECIO" n "DISKW EMAIL (FINIS"
"IDIFREE DD(EMAIL)"
end
exit 0

```

This is a sample job that can be used to invoke this exit. Change <dump-data-set-name> to the dump data set name for which an entry should be created.

```

//REGDUMP JOB MSGCLASS=X,NOTIFY=&SYSUID
//RUNFA EXEC PGM=IDIDA,
// PARM='DumpDSN(<dump-data-set-name>)'
//IDIEXEC DD DISP=SHR,DSN=IDI.SIDIEXEC
//IDIOPTS DD *
    Exits(Format(REXX(IDISUFM5)))
/*

```

## Additional sample Formatting user exits

### IDISUFM6

Sample Fault Analyzer Formatting user exit to format LE CAA and CIB.

### IDISUFM7

Sample Fault Analyzer Formatting user exit to display Name/Token values.

## End Processing user exit

The following describes the End Processing user exit.

### Purpose

This exit can be used to control post-analysis actions taken by Fault Analyzer:

- **History file selection.**

Following analysis, detailed information about the fault is available which might not have been available at the time of calling any Analysis Control user exits. Based on this, an End Processing user exit can choose to change the history file to be used by modifying the ENV.IDIHIST data area field.

Because information about duplicate faults depend on the history file that is selected, the End Processing user exit is invoked repeatedly until the history file name is no longer changed. On each reinvocation, duplicate fault information is updated for the history file specified on the previous invocation.

If an invalid history file is provided in the ENV.IDIHIST field, then no reinvocation of the End Processing user exit occurs, and the history file that was current when the End Processing user exit was last invoked is used.

- **Duplicate fault determination.**

By default, Fault Analyzer deems a fault a duplicate of another fault in the same history file if the characteristics of the faults match and they occurred within the

## End Processing user exit

number of hours in effect for the NoDup(NORMAL(*hours*)) option of each other (see “NoDup” on page 482 for details). The End Processing user exit permits an installation to apply a different time interval for the determination of duplicate faults to the one in effect due to the NoDup option.

If a duplicate fault was found based on the fault characteristics alone, then the following information is provided:

- The field EPC.MINUTES\_SINCE\_LAST\_DUP is initialized to the number of minutes elapsed since recording of the last duplicate fault. The value is in the range 0 - 99999 (values greater than the limit of this field are all presented as the maximum value, 99999). If no value is provided, no duplicate fault was found.
- The field EPC.DUPLICATE\_COUNT shows the total number of times that a fault was deemed a duplicate, not including the current fault.

This total is determined by accumulating all instances of duplicate recorded faults, in the same history file, going back as far as the NoDup(Normal(...)) time period in effect. Any recorded faults encountered whose duplicate criteria match the current fault accounts for one instance, and if duplicates have been recorded against the fault, the duplicate count of that fault is also added.

- The field ENVFAULT\_ID identifies the recorded duplicate fault by its fault ID.
- The fields ENV.DUP\_DATE and ENV.DUP\_TIME identify the date and time of most recent duplicate fault.
- The fields ENV.ORIGINAL\_DATE and ENV.ORIGINAL\_TIME identify the date and time when the original fault was recorded. These fields provide the user with the ability to determine duplicates on the basis of time since the original fault, as opposed to the last duplicate fault, if so desired.

If both the fault characteristics matched and the elapsed time did not exceed the number of hours in effect for the NoDup(NORMAL(*hours*)) option, the field EPC.IS\_DUPLICATE is initialized to 'Y'. However, the final determination of whether the fault is a duplicate or not resides with the End Processing user exit. If the returned value of this field is 'Y', the last recorded duplicate fault's (if any) duplicate count is incremented by one and message IDI0044I is issued.

**Note:** CICS or IMS fast duplicate fault suppression cannot be controlled using this exit. Only the NoDup(CICSFAST(...)) or the NoDup(ImageFast(...)) option (see “NoDup” on page 482) can be used to affect the determination of these types of duplicates. However, if a fault occurring under CICS or IMS is not deemed a fast duplicate, then it is still considered for the normal type of duplicate suppression based on existing entries in the target history file and the NoDup(NORMAL(*hours*)) option in effect.

- **History file updates.**

By default, Fault Analyzer suppresses the entire fault entry, including the minidump, if the current fault is found to be a duplicate of a previously recorded fault in the same history file. However, the minidump might also be suppressed if its size exceeds the limit imposed by the MaxMinidumpPages option in effect.

The possible reasons for suppression in the following fields:

- The ENV.MINIDUMP\_PAGES field, containing the size of the minidump as a number of 4K pages.
- The EPC.IS\_DUPLICATE field (see “Duplicate fault determination” above).

The suppression intent by Fault Analyzer is indicated by the initialization of the following fields, both of which can be overridden by the End Processing user exit:

- The EPC.SUPPRESS\_MINIDUMP field. If set to 'Y', no minidump is written to the history file. If set to 'N', the minidump is written regardless of its size.
- The EPC.SUPPRESSFAULT\_ENTRY field. If set to 'Y', no recording of the current fault is made in the history file (including the minidump). If set to 'N', fault recording is performed and the minidump can be written subject to the EPC.SUPPRESS\_MINIDUMP field.

- **Dump suppression.**

The End Processing user exit can set the EPC.SUPPRESS\_DUMP field to 'Y' if dumps should be suppressed, or to 'N' if they should be permitted to be taken. Refer to "Dump suppression" on page 15 for general information about dump suppression.

### **When invoked**

This exit is invoked on completion of real-time analysis, prior to updating the history file.

### **Parameters**

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the End Processing user exit.

**REXX:** Two stems are available to the exit:

- ENV.  
Contains defined symbols for all fields in the ENV data area (see "ENV - Common exit environment information" on page 513).
- EPC.  
Contains defined symbols for all fields in the EPC data area (see "EPC - End Processing user exit parameter list" on page 521).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area (see "ENV - Common exit environment information" on page 513).
- 31-bit EPC address in word 2.  
Address of an EPC data area (see "EPC - End Processing user exit parameter list" on page 521).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### **Example**

The following is an example of an End Processing user exit that is written in REXX.

## End Processing user exit

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if EPC.VERSION <> 1 then
  say 'Note: EPC data area version change - field usage review required!'
if EPC.MINUTES_SINCE_LAST_DUP ~= ' ' & EPC.MINUTES_SINCE_LAST_DUP < 48*60 then do
  /* Use 48 hours as the duplicate fault threshold */
  EPC.IS_DUPLICATE = 'Y'
  EPC.SUPPRESS_FAULT_ENTRY = 'Y'
end
EPC.SUPPRESS_DUMP = 'N' /* Always permit dumps to be taken */
exit 0
```

---

Figure 150. Sample REXX End Processing user exit

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(END(REXX(ABC)))
```

## End Processing user exit (Fault entry refresh)

The following describes the fault entry refresh End Processing user exit.

### Purpose

This exit can be used to control history file updates.

By default, Fault Analyzer suppresses the minidump if its size exceeds the limit imposed by the MaxMinidumpPages option in effect.

The minidump size is available in ENV.MINIDUMP\_PAGES field as a number of 4K pages.

The suppression intent by Fault Analyzer is indicated by the initialization of the EPC.SUPPRESS\_MINIDUMP data area field, which can be overridden by the End Processing user exit. If set to 'Y', no minidump is written to the history file. If set to 'N', the minidump is written regardless of its size.

The End Processing user exit can choose to suppress the refresh of the entire fault entry using the EPC.SUPPRESS\_FAULT\_ENTRY data area field. If set to 'Y', no refresh of the current fault is performed (including the minidump). If set to 'N', the refresh is performed and the minidump can be written subject to the SUPPRESS\_MINIDUMP field.

### When invoked

This exit is invoked on completion of batch reanalysis, prior to updating the history file.

### Parameters

See "Parameters" on page 395.

### Example

The following is an example of a fault entry refresh End Processing user exit that is written in REXX.

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if EPC.VERSION <> 1 then
  say 'Note: EPC data area version change - field usage review required!'
EPC.SUPPRESS_MINIDUMP = 'Y' /* Always suppress the minidump */
exit 0
```

Figure 151. Sample fault entry refresh REXX End Processing user exit

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
RefreshExits(END(REXX(ABC)))
```

## Notification user exit

The following describes the Notification user exit.

### Purpose

This exit can be used to provide installation-specific notification about the recording of a fault in a history file, or the occurrence of a duplicate of an earlier fault.

The reason for invoking the exit is identified in the NFY data area<sup>15</sup> NFYTYPE field as one of the following:

**C** Fault created.

The name of the history file in which the fault was recorded is provided in the ENV.IDIHIST data area field, and the fault ID assigned to the recorded fault is available in the ENVFAULT\_ID data area field.

Fault Analyzer issues message IDI0003I to indicate the assigned fault ID and history file.

A copy of the synopsis section of the real-time report is available in the NFY.SYNOPSIS data area field. Each line of the synopsis is delimited by a new-line character (X'15'). Refer to the NFY data area for extra details regarding this field.

**R** Recovery fault recording

This value indicates a fault that was created as a result of the recovery fault recording feature of Fault Analyzer. (For more information about this feature, see "Recovery fault recording" on page 30.)

The name of the history file in which the fault was recorded is provided in the ENV.IDIHIST data area field, and the fault ID assigned to the recorded fault is available in the ENVFAULT\_ID data area field.

Fault Analyzer issues message IDI0126I to indicate the assigned fault ID and history file.

**N** Normal duplicate.

---

15. See "Parameters" on page 398 for references to the NFY data area.

## Notification user exit

This value indicates that the NoDup(NORMAL) option criteria matched for the current fault, and that no history file fault entry is therefore written. (For details about NoDup(NORMAL), see “NoDup” on page 482.)

The original history file name and fault ID are provided in the ENV.IDIHIST and ENVFAULT\_ID data area fields.

The DUPCOUNT field is set to 1.

- F Fast duplicate (CICS).

This value indicates that the NoDup(CICSFAST) option criteria matched for the current fault, and that no analysis was therefore performed. (For details about NoDup(CICSFAST), see “NoDup” on page 482.)

If available, the original history file name and fault ID are provided in the ENV.IDIHIST and ENVFAULT\_ID data area fields.

The DUPCOUNT field is set to the number of duplicate occurrences for the 30-second recording period.

### When invoked

This exit is invoked after Fault Analyzer has finished the recording of a fault in the history file.

### Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Notification user exit.

**REXX:** Two stems are available to the exit:

- ENV.  
Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).
- NFY.  
Contains defined symbols for all fields in the NFY data area (see “NFY - Notification user exit parameter list” on page 524).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area (see “ENV - Common exit environment information” on page 513).
- 31-bit NFY address in word 2.  
Address of an NFY data area (see “NFY - Notification user exit parameter list” on page 524).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

**Calling a non-REXX logging routine from REXX:** For a Notification user exit that is written in REXX that, for example, calls an external logging routine that is not

written in REXX, two extra stem variables are provided. The stem variables can be used to pass all of the available data area values to the external routine. These stem variables are:

```
ENV.RECORD
NFY.RECORD
```

Each of these contain the entire ENV or NFY data area respectively, in a single REXX variable. Neither exist as fields in the respective data areas for non-REXX exits, as they each represent the entire data area, which is already provided.

By using these variables in an argument list for an external non-REXX routine, the REXX exit need not be concerned with changes that might occur to these data areas in the future.

The external routine should use the language-dependent data area mappings provided for access to any data values (see "Load module exits" on page 362). All values in the ENV.RECORD and NFY.RECORD variables are considered read only and cannot be updated by the external routine. If updates are required, these must be made in the appropriate data field stem variables by the REXX exit itself.

### Example 1

The following is an example of a Notification user exit that is written in REXX. It issues a TSO SEND message.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SEND command via TSO batch job */
queue "//NOTIFY JOB MSGCLASS=Z"
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD SYSOUT=**"
queue "//SYSTSIN DD *"
/* Split the TSO SEND command over three data records that must
   each be padded with blanks to 80 bytes */
rec = "SEND 'Fault ID' ENVFAULT_ID assigned in history file -"
queue left(rec,80)
rec = strip(ENV.IDIHIST)||" -"
queue left(rec,80)
rec = "USER("||strip(ENV.USER_ID)||") LOGON"
queue left(rec,80)
queue '/*'
/* 'Submit' the stacked TSO batch job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
address mvs "EXECIO" n "DISKW DD1 (FINIS"
"IDIFREE DD(DD1)"
exit 0
```

---

*Figure 152. Sample REXX Notification user exit 1*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(NOTIFY(REXX(ABC)))
```

## Notification user exit

### Example 2

The following is an example of a Notification user exit that is written in REXX. It sends an e-mail message via SMTP.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SMTP message via SMTP batch interface */
user = strip(ENV.USER_ID)
queue "he1o pthmvs8.au.ibm.com"
queue "mail from:<"user"@pthmvs8.au.ibm.com>"
queue "rcpt to:<"user"@au1.ibm.com>" 
queue "data"
queue "Date: " date('N') time('C')
queue "From:<"user"@pthmvs8.au.ibm.com>" 
queue "To:<"user"@au1.ibm.com>" 
queue "Subject: Batch job "strip(ENV.JOB_NAME)" abend "ENV.ABEND_CODE
queue ""
queue "Fault ID "ENVFAULT_ID" assigned in history file"
queue strip(ENV.IDIHIST) for job "ENV.JOB_NAME"
queue "program "ENVEXEC_PGM_NAME" module "ENV.ABEND_MODULE_NAME"."
queue ""
n = queued()
"IDIALLOC DD(DD1) SYSOUT(A) DEST(PTHMVS8.SMTP)"
address mvs "EXECIO" n "DISKW DD1 (FINIS"
"IDIFREE DD(DD1)"
exit
```

Figure 153. Sample SMTP REXX Notification user exit 2

The MVS TCP/IP SMTP environment must be available to your system for this exit to work.

The previous sample exit has been tested using the batch interface of the IBM SMTP server provided with the IBM Communications Server product.

Successful delivery of the sample message is dependent on configuration of the IBM CS TCP/IP service and SMTP server, and a suitable TCP/IP network infrastructure being available on the system running the exit.

Factors such as the presence of firewalls, security software, and message filtering software on intermediate network nodes might affect successful delivery of the message.

If the above sample exit existed as member NOTIFY1 in data set TEST.EXEC.PDS, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(TEST.EXEC.PDS))
Exits(NOTIFY(REXX(NOTIFY1)))
```

### Example 3

The following is an example of a Notification user exit that is written in REXX. It extracts and shows all lines of the captured real-time synopsis.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Show synopsis */
rest = NFY.SYNOPSIS
do while rest<>''
  parse var rest nextline '15'x rest
  say nextline
end
exit 0
```

---

*Figure 154. Sample SMTP REXX Notification user exit 3*

This example can, for example, be combined with example 2 above, to include the synopsis in an e-mail message.

If the above sample exit existed as member FRED in data set TEST.EXEC.PDS, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(TEST.EXEC.PDS))
Exits(NOTIFY(REXX(FRED)))
```

#### Example 4

The following is an example of a Notification user exit that is written in REXX and can be used to submit a batch reanalysis job to generate a saved report for a DeferredReport fault entry.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
queue "//GENREP JOB MSGCLASS=X"
queue "//FA EXEC PGM=IDIDA,"
queue "// PARM=('//FAULTID("ENV.FAULT_ID")','"
queue "//      'GenerateSavedReport',"
queue "//      )"
queue "//IDIHIST DD DISP=SHR,DSN="ENV.IDIHIST
queue "//SYSPRINT DD SYSOUT=*" 
/* 'Submit' the stacked batch reanalysis job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
address mvs "EXECIO" n "DISKW DD1 (FINIS"
"IDIFREE DD(DD1)"
exit 0
```

---

*Figure 155. Sample REXX Notification user exit 4*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(NOTIFY(REXX(ABC)))
```

## **Notification user exit (MVS SVC Dump registration)**

The following describes the dump registration Notification user exit.

### **Purpose**

This exit can be used to provide installation-specific notification about the recording of an SVC dump fault entry in a history file.

The name of the history file in which the fault was recorded is provided in the ENV.IDIHIST data area field, and the fault ID assigned to the recorded fault is available in the ENVFAULT\_ID data area field.

Fault Analyzer issues message IDI0003I to indicate the assigned fault ID and history file.

### **When invoked**

This exit is invoked after Fault Analyzer has finished the registration of an MVS SVC dump fault entry in the history file.

### **Parameters**

See "Parameters" on page 398.

### **Example**

The following is an example of a dump registration Notification user exit that is written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SEND command via TSO batch job */
queue "//NOTIFY JOB MSGCLASS=Z"
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD SYSOUT=*" 
queue "//SYSTSIN DD *"
/* Split the TSO SEND command over three data records that must
   each be padded with blanks to 80 bytes */
rec = "SEND 'Fault ID' ENVFAULT_ID \"assigned in history file -\""
queue left(rec,80)
rec = strip(ENV.IDIHIST)||"'" -"
queue left(rec,80)
rec = "USER(FRED) LOGON"
queue left(rec,80)
queue '/*'
/* 'Submit' the stacked TSO batch job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
address mvs "EXECIO" n "DISKW DD1 (FINIS"
"IDIFREE DD(DD1)"
exit 0
```

---

*Figure 156. Sample REXX dump registration Notification user exit*

Note that, unlike the normal End Processing user exit, no user ID is available in the ENV data area.

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following options in either the IDICNF00 configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(X.Y.Z))  
DumpRegistrationExits(NOTIFY(REXX(ABC)))
```

## Descriptions of IDIUTIL batch utility user exit types

The following provides descriptions of each IDIUTIL batch utility user exit type available for use with Fault Analyzer.

### IDIUTIL Import user exit

The following describes the IDIUTIL Import user exit.

#### Purpose

This exit can be used to control the import of a fault entry during history file management using the IDIUTIL batch utility with the IMPORT control statement. This control is provided by setting the UTL.PERFORM\_ACTION data area field<sup>16</sup> to 'Y' if the entry should be imported, or to 'N' if not. By default, the field UTL.PERFORM\_ACTION is always set to 'Y' before invoking the exit.

When an entry has been successfully imported into the target history file, it is deleted from the source history file.

#### When invoked

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the IMPORT control statement.

#### Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Notification user exit.

**REXX:** Two stems are available to the exit:

- ENV.  
Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).
- UTL.  
Contains defined symbols for all fields in the UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area (see “ENV - Common exit environment information” on page 513).
- 31-bit UTL address in word 2.

16. See “Parameters” for references to the UTL data area.

## **IDIUTIL Import user exit**

Address of a UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### **Example**

The following is an example of an IDIUTIL Import user exit that is written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 1 then
  say 'Note: UTL data area version change - field usage review required!'
UTL.PERFORM_ACTION = 'Y' /* Import current entry (default) */
```

---

*Figure 157. Sample REXX IDIUTIL Import user exit*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the JCL DD statement

```
//IDIEXEC DD DISP=SHR,DSN=X.Y.Z
```

and the IDIUTIL batch utility control statement

```
Exits(IMPORT(REXX(ABC)))
```

in your IDIUTIL batch utility history file management job would cause the exit to be invoked.

## **IDIUTIL Delete user exit**

The following describes the IDIUTIL Delete user exit.

### **Purpose**

This exit can be used to control the deletion of a fault entry during history file management using the IDIUTIL batch utility with the DELETE control statement (for details, see “DELETE control statement” on page 339). This control is provided by setting the data area field UTL.PERFORM\_ACTION<sup>17</sup> to 'Y' if the entry should be deleted, or to 'N' if not. The field UTL.PERFORM\_ACTION is set to 'Y' before invoking the exit, except when a fault entry is locked. In this case, when ENV.LOCK\_FLAG is not blank, the UTL.PERFORM\_ACTION flag is set to 'N'.

The fault entries for which the user exit is invoked are those that match the specified DELETE control statement criteria.

### **When invoked**

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the DELETE control statement.

### **Parameters**

How parameters are passed to the exit depends on the exit type, REXX or load module.

---

17. See “Parameters” for references to the UTL data area.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Notification user exit.

**REXX:** Two stems are available to the exit:

- ENV.

Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).

- UTL.

Contains defined symbols for all fields in the UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

Address of an ENV data area (see “ENV - Common exit environment information” on page 513).

- 31-bit UTL address in word 2.

Address of a UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### Example

The following is an example of an IDIUTIL Delete user exit that is written in REXX.

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 1 then
  say 'Note: UTL data area version change - field usage review required!'
UTL.PERFORM_ACTION = 'Y' /* Delete current entry */
```

Figure 158. Sample REXX IDIUTIL Delete user exit

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the JCL DD statement

```
//IDIEXEC DD DISP=SHR,DSN=X.Y.Z
```

and the IDIUTIL batch utility control statement

```
Exits(DELETE(REXX(ABC)))
```

in your IDIUTIL batch utility history file management job would cause the exit to be invoked.

## IDIUTIL ListHF user exit

The following describes the IDIUTIL ListHF user exit.

### Purpose

This exit can be used to control the listing of a fault entry during history file management using the IDIUTIL batch utility with the LISTHF control statement (for details, see “LISTHF control statement” on page 338). This control is provided by setting the data area field UTL.PERFORM\_ACTION<sup>18</sup> to ‘Y’ if the entry should be listed, or to ‘N’ if not. The field UTL.PERFORM\_ACTION is set to ‘Y’ before invoking the exit.

The fault entries for which the user exit is invoked are those that match the specified LISTHF control statement criteria.

### When invoked

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the LISTHF control statement.

### Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Notification user exit.

**REXX:** Two stems are available to the exit:

- ENV.  
Contains defined symbols for all fields in the ENV data area (see “ENV - Common exit environment information” on page 513).
- UTL.  
Contains defined symbols for all fields in the UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

**Load module:** At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area (see “ENV - Common exit environment information” on page 513).
- 31-bit UTL address in word 2.  
Address of a UTL data area (see “UTL - IDIUTIL Batch Utility user exit parameter list” on page 530).

**Note:** The high-order bit is on to indicate that this parameter is the last parameter passed.

### Example 1

The following is an example of an IDIUTIL ListHF user exit that is written in REXX.

18. See “Parameters” for references to the UTL data area.

---

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 1 then
  say 'Note: UTL data area version change - field usage review required!'
UTL.PERFORM_ACTION = 'Y' /* List current entry */
```

---

*Figure 159. Sample REXX IDIUTIL ListHF user exit 1*

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the JCL DD statement

```
//IDIEEXEC DD DISP=SHR,DSN=X.Y.Z
```

and the IDIUTIL batch utility control statement

```
Exits(LISTHF(REXX(ABC)))
```

in your IDIUTIL batch utility history file management job would cause the exit to be invoked.

## Example 2

The following is an example of an IDIUTIL ListHF user exit that is written in REXX.

This sample exit shows how one might write a customized report, as well as a comma-delimited file that can be used as input to a spreadsheet application.

In addition to the fields used, any other fields that are available from the ENV or UTL data areas can be included.

The report that is written by the provided sample includes the columns:

- Fault ID
- Date
- Time
- Lock
- Username
- User Title
- CPU Sec

See “Available columns” on page 45 for explanations of the column data, except for “CPU Sec”, which is the total CPU time used by Fault Analyzer based on ENV.CPU\_HSECONDS. See 518 for more information about this value.

The customized report is written to DDname MYREP. A MYREP DD statement must be included in the IDIUTIL job that invokes this exit, for example:

```
//MYREP DD SYSOUT=*
```

The comma-delimited file is written to DDname COMMA. A COMMA DD statement must be included in the IDIUTIL job that invokes this exit, for example:

```
//COMMA DD SYSOUT=*
```

The persistent user field, ENV.USER\_1, is used to record the fact that the report header has been written.

## IDIUTIL ListHF user exit

---

```
/* First ensure that the current data area versions match the */  
/* versions as at the time of coding the exit. */  
If ENV.VERSION <> 5 Then  
  Say 'Note: ENV data area version change - field usage review',  
    'required!'  
If UTL.VERSION <> 2 then  
  Say 'Note: UTL data area version change - field usage review',  
    'required!'  
If ENV.USER_1=''' Then Do  
  /* Write report header */  
  out.1="Fault ID Date      Time      Lock Username",  
    "User Title                      CPU Sec"  
  out.2="----- ----- ----- -----"  
  "----- ----- ----- -----"  
  ADDRESS MVS "EXECIO 2 DISKW MYREP (STEM out."  
  /* Write comma-delimited file header */  
  out.1="Fault ID,Date,Time,Lock,Username,User Title,CPU Sec"  
  ADDRESS MVS "EXECIO 1 DISKW COMMA (STEM out."  
  ENV.USER_1='done'      /* Flag header done.  
End  
/* The fault ID value is placed right-aligned in a work field. */  
fault_id=COPIES(' ',8-length(ENVFAULT_ID))||ENVFAULT_ID  
/* The following lines use the REXX INSERT command to ensure that the */  
/* work fields for each value are padded with blanks to fit the */  
/* report column width. */  
/* For information about the maximum width of any field, refer to the */  
/* User's Guide and Reference "Data Areas" chapter. */  
abend_date=INSERT(ENVABEND_DATE,'',,10)  
abend_time=INSERT(ENVABEND_TIME,'',,8)  
lock_flag =INSERT(ENVLOCK_FLAG,'',,4)  
user_name =INSERT(ENVUSER_NAME,'',,8)  
user_title=INSERT(ENVUSER_TITLE,'',,40)  
/* If available, the CPU time in 1/100s of a second is changed to a */  
/* number of seconds with two decimal digits. */  
if ENV.CPU_HSECONDS=''' then cpu_sec=''  
else cpu_sec=FORMAT(ENV.CPU_HSECONDS/100,4,2)  
/* Write report line for this fault entry. */  
out.1=fault_id abend_date abend_time lock_flag user_name user_title,  
  cpu_sec  
ADDRESS MVS "EXECIO 1 DISKW MYREP (STEM out."  
/* Write comma-delimited line for this fault entry. */  
out.1=fault_id","abend_date","abend_time","lock_flag","user_name,  
  ","user_title","cpu_sec  
ADDRESS MVS "EXECIO 1 DISKW COMMA (STEM out."  
UTL.PERFORM_ACTION='N' /* Optionally, suppress the standard report. */  
Exit 0
```

---

Figure 160. Sample REXX IDIUTIL ListHF user exit 2

The above sample exit is provided as member IDISUTL1 in the IDI.SIDISAM1 data set.

The following JCL could be used to run the sample:

```
//IDIUTIL JOB parms  
//RUNUTIL EXEC PGM=IDIUTIL  
//SYSPRINT DD SYSOUT=*<br/>  
//MYREP DD SYSOUT=*<br/>  
//COMMA DD SYSOUT=*<br/>  
//IDITRACE DD SYSOUT=*(Optional)<br/>  
//IDIEXEC DD DISP=SHR,DSN=IDI.SIDISAM1  
//SYSIN DD *
```

```
Exits(LISTHF(REXX(IDISUTL1)))
FILES(my.histfile)
LISTHF
/*
```

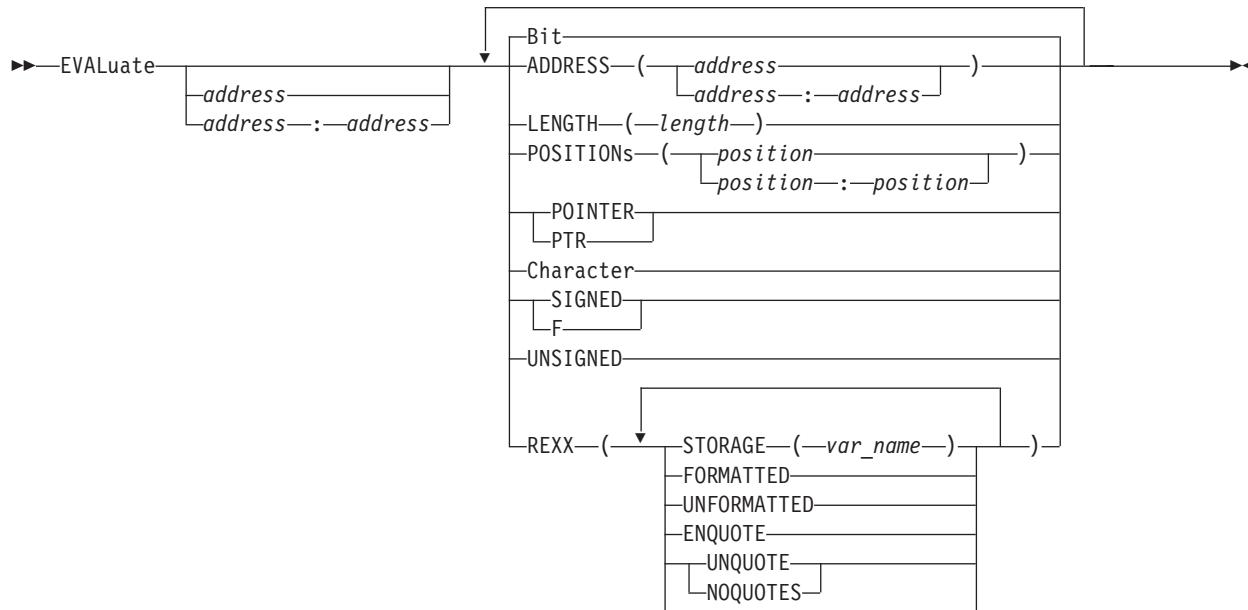
## User exit REXX commands

Fault Analyzer provides the following special REXX commands for use in user exits written in REXX. All of these commands are available in the default REXX environment, FAULTA. Generally, if another environment has been made current using the REXX ADDRESS instruction, then it is necessary to precede the Fault Analyzer REXX commands with ADDRESS FAULTA. However, the EVALUATE, LIST, and NOTE commands are available with both ADDRESS FAULTA and with ADDRESS IPCS for compatibility with REXX exits for the IPCS environment.

### Evaluate command

The Evaluate command can be used to by a Formatting user exit to obtain storage from the analyzed fault environment.

#### Syntax



#### Parameters

**address**

**start:end**

Specifies either the start address, or an address range, as a positional parameter. This address is an alternative to using the ADDRESS keyword. See *ADDRESS* below for details about valid syntax.

## Evaluate command

**ADDRESS(address)**

**ADDRESS(address:address)**

Specifies the address of the storage to be returned. Specify as either a single address or as an address range.

The address parameter is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93. The address may optionally followed by a period (for example, "000176C0.".).

**LENGTH(length)**

Specifies the number of bytes to be returned. Specify as a hexadecimal value (indicated by X'...') or as a decimal value (no indication required). If an address range is specified, then specification of LENGTH is ignored.

**POSITIONs(position)**

**POSITIONs(position:position)**

Specifies the offset from the start address to the first byte of storage to be returned. Specify as either a single offset or as an offset range. Both POSITION and POSITIONS are accepted.

All offsets must be either hexadecimal (indicated by X'...') or signed decimal (optionally indicated by F'...').

**Bit** Specifies that data is to be returned in hexadecimal format.

This parameter is only valid if FORMATTED is in effect.

**POINTER**

**PTR** Specifies that data is to be returned in hexadecimal format.

The valid LENGTH is 1 - 4 bytes. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

**Character**

Specifies that data is to be returned in character string format with non-printable characters replaced by periods. Further editing is determined by the specification of ENQUOTE, UNQUOTE, NOQUOTES, or NOBLANKS parameters.

This parameter is only valid if FORMATTED is in effect.

**SIGNED**

**F** Specifies that data is to be returned in signed decimal format. Leading zeroes are removed and a minus sign is supplied for negative integers.

The valid LENGTH is either 2 or 4 bytes. If LENGTH is 1 or 3, then it is changed to 2. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

**UNSIGNED**

Specifies that data is to be returned in unsigned decimal format. Leading zeroes are removed.

The valid LENGTH is 1 - 4 bytes. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

**REXX(...)**

Specifies that the storage is to be returned in a REXX variable. This parameter is required.

**STORAGE(var\_name)**

Specifies the name of the REXX variable which is to receive the storage.  
This parameter is required.

**FORMATTED**

Specifies that storage is to be returned formatted. This value is the default.

**UNFORMATTED**

Specifies that storage is to be returned in raw hex format (one byte returned for each byte of storage).

**ENQUOTE**

Specifies that one leading and one trailing quote are to be added to the returned character string, and that any apostrophes found in the string are to be paired.

This parameter is only valid if character data is being returned.

**UNQUOTE****NOQUOTES**

Specifies that all apostrophes (X'7D') in the returned character string are to be replaced by periods.

This parameter is only valid if character data is being returned.

**NOBLANKS**

Specifies that all blanks in the returned character string are to be replaced by periods.

This parameter is only valid if character data is being returned.

If a parameter is specified multiple times, then only the last specification takes effect.

**Return codes**

The Evaluate command provides the following return codes:

- 0** Storage was obtained successfully.
- 4** The requested data length was modified. An explanation is written to the IDITRACE DDname.
- 12** Command syntax error or storage not available. An explanation of the error is written to the IDITRACE DDname.

**Example**

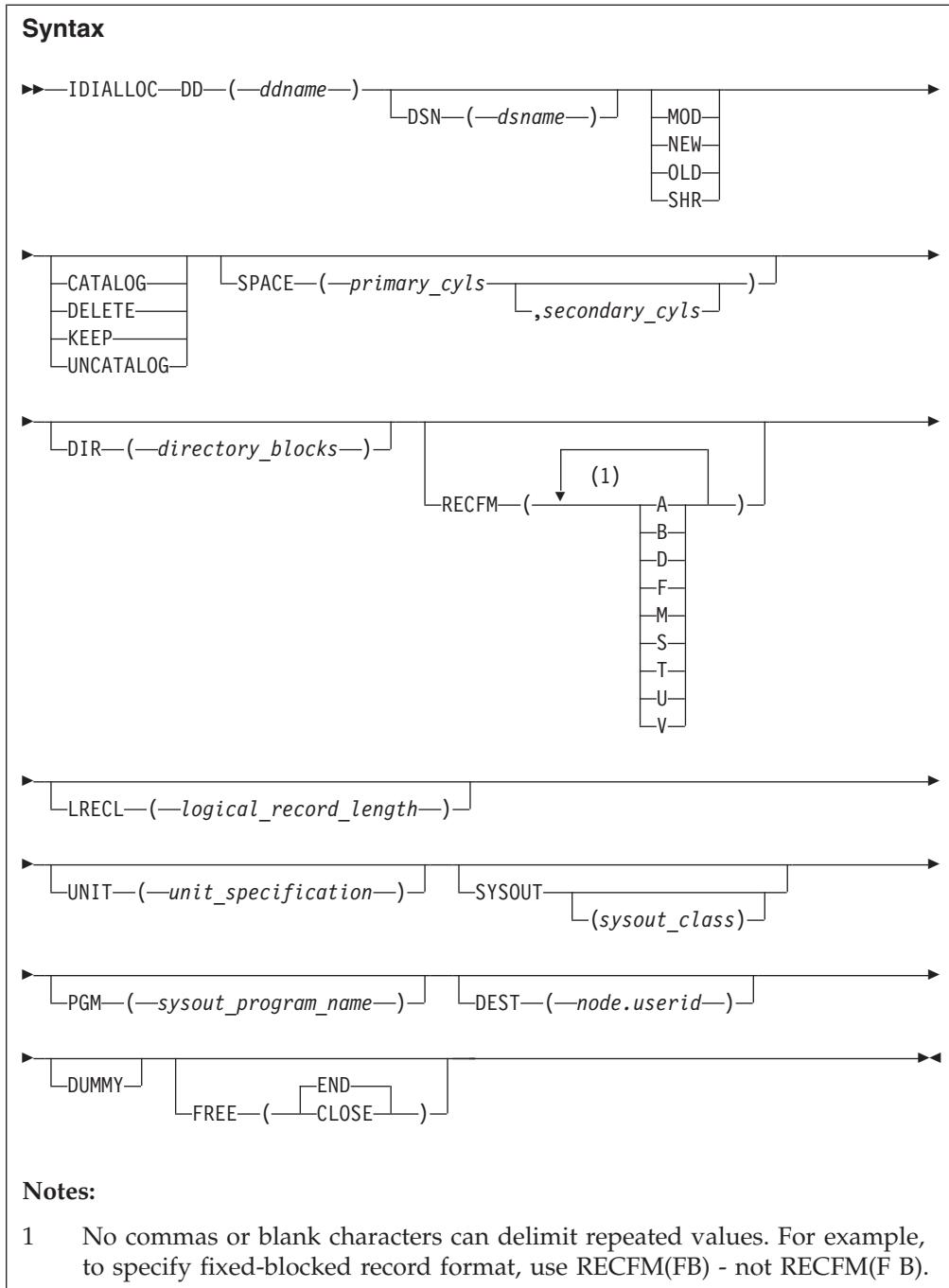
```
/* REXX */
"EVALUATE 0 LENGTH(128) REXX(STORAGE(x))"
if RC = 0 then say 'Storage at address 0 =' x
```

*Figure 161. Evaluate command example*

**IDIALLOC command**

The IDIALLOC command can be used to perform dynamic allocation of data sets to DDnames.

## IDIALLOC command



Operands:

- DD** DDname to be associated with data set. This DDname is always required.
- DSN** Name of data set to be allocated.
- MOD** Additions are to be made to data set.
- NEW** Data set is to be created.
- OLD** Data set exists and exclusive control is required.
- SHR** Data set exists but exclusive control is not required.

**CATALOG**

Data set is to be cataloged.

**DELETE**

Data set is to be deleted when freed.

**KEEP** Data set is to be kept when freed.

**UNCATALOG**

Data set is to be uncataloged.

**SPACE**

Primary space and increment as number of cylinders.

**DIR** Number of directory blocks required.

**RECFM**

Record format:

- A** ASA printer characters
- B** Blocked
- D** Variable length ASCII records
- F** Fixed
- M** Machine control character
- S** Standard blocks or spanned
- T** Track overflow
- U** Undefined
- V** Variable

**LRECL**

Logical record length (0 to 32760 value).

**UNIT** Device type to which a file or data set is to be allocated.

**SYSOUT**

Data set is to be system output data set. The class can optionally be specified as a single character.

**PGM** SYSOUT program name.

**DEST** DEST node and user ID.

Just the user ID can be used for local destinations.

**DUMMY**

Allocate dummy data set.

**FREE** Deallocation specification:

**CLOSE**

Requests that the system deallocate the data set when it is closed.

**END** Requests that the system deallocate the data set at the end of the last step that references the data set. This value is the default.

The following syntax rules apply:

- DSN is mutually exclusive with the following parameters:
  - PGM
  - DEST
- DEST is mutually exclusive with the PGM parameter.

## IDIALLOC command

- SYSOUT is mutually exclusive with the following parameters:
  - OLD
  - MOD
  - SHR
  - NEW
- The following parameters require that the DSN parameter is also specified:
  - SPACE
  - DIR
  - UNIT
- The following parameters require that the SYSOUT parameter is also specified:
  - PGM
  - DEST

**Note:** No automatic prefixing of user ID to data set names are performed by Fault Analyzer for this command. All data set names must be fully qualified and specified without quotes.

### Return codes

The IDIALLOC command provides the following return codes:

- 0 The allocation was successful. If a member of a PDS(E) was allocated, the member exists and can be opened for read.
- 1 The allocation was successful. However, a non-existing member of a PDS(E) was allocated which cannot be opened for read. If the member is opened for read, a system abend S013 occurs.
- 4 Allocation failed. An explanation of the error is written to the IDITRACE DDname.
- 8 Command syntax error. An explanation of the error is written to the IDITRACE DDname.

### Example

---

```
/* REXX */

/* Allocate an existing data set to DDname DD1 */
"IDIALLOC DD(DD1) DSN(FRED.LISTING) SHR"
if RC = 0 then say 'Success!'

/* Allocate a temporary sequential work data set to DDname DD2 */
"IDIALLOC DD(DD2) DSN(FRED.SEQ) NEW DELETE SPACE(2,3) UNIT(SYSALLDA) ",
   "RECFM(FB) LRECL(80)"

/* Allocate a new partitioned data set to DDname DD3 */
"IDIALLOC DD(DD3) DSN(FRED.PDS) NEW CATALOG SPACE(2) UNIT(SYSALLDA) ",
   "DIR(5) RECFM(VBA) LRECL(137)"

/* Allocate default JES spool data set to DDname DD4 */
"IDIALLOC DD(DD4) SYSOUT"

/* Allocate internal reader for submission of job to DDname DD5 */
"IDIALLOC DD(DD5) SYSOUT PGM(INTRDR)"
```

---

Figure 162. IDIALLOC command example

## IDIDDTEST command

The IDIDDTEST command can be used to test if a DDname is allocated.

### Syntax

```
►—IDIDDTEST—DD—(—ddname—)—————►
```

### Return codes

The IDIDDTEST command provides the following return codes:

- 0**      The specified DDname is allocated.
- 4**      The specified DDname is not allocated.
- 8**      Command syntax error. An explanation of the error is written to the IDITRACE DDname.

### Example

---

```
/* REXX */

/* Test if the DDname DD1 is allocated */
"IDIDDTEST DD(DD1)"
if RC = 0 then say 'DD1 is allocated'
```

---

Figure 163. IDIDDTEST command example

## IDIDSECTdsn command

The IDIDSECTdsn command can be used to query or modify the IDIDSECT data set concatenation (for details about this DDname, see “DataSets” on page 457) to ensure that the correct DSECT mapping is used for a given version of a product, for example CICS.

### Syntax

```
►—IDIDSECTDSN—[GET—(—var_name—)]—[SET—(—var_name—)]—————►
```

where:

#### GET(*var\_name*)

Specifies that the current IDIDSECT data set concatenation is returned in the REXX variable *var\_name*. The data set names in the returned list are blank delimited.

#### SET(*var\_name*)

Specifies the name of a REXX variable containing the list of data set names that replace the current IDIDSECT concatenation. Multiple data set names must be separated by one or more blanks.

## IDIDSECTdsn command

### Return codes

The IDIDSECTdsn command provides the following return codes:

- 0 Successful completion.
- 4 Command syntax error. An explanation of the error is written to the IDITRACE DDname.

### Example

```
/* REXX */  
  
/* Place MY.DSECTS data set first in the IDIDSECT concatenation */  
"IDIDSECTdsn GET(dsnlist)"  
if RC = 0 then say 'Current IDIDSECT concatenation:' dsnlist  
else exit 4  
dsnlist = 'MY.DSECTS' dsnlist  
"IDIDSECTdsn SET(dsnlist)"  
if RC = 0 then say 'IDIDSECT concatenation changed to:' dsnlist
```

Figure 164. IDIDSECTdsn command example

## IDIEventInfo command

The IDIEventInfo command can be used to by a Formatting user exit to obtain information about any event in the current fault.

Since the UFM data area only contains one set of fields for PSW, registers, and so on, it is necessary to use the IDIEventInfo command to change the values in the UFM data area from the values representing one event to those of another.

### Syntax

```
►►—IDIEventInfo— [var_name]—►►
```

### Parameters

#### var\_name

Specifies the name of a REXX variable containing the event number for which information is to be retrieved.

If a variable containing the desired event number is not provided in *var\_name*, then UFM.EVENT\_NO is used instead.

The information retrieved is placed in the UFM data area.

### Return codes

The IDIEventInfo command provides the following return codes:

- 0 Information was retrieved successfully.
- 4 No information available for the specified event number. An explanation of the error is written to the IDITRACE DDname.

- 8        Command syntax error. An explanation of the error is written to the IDITRACE DDname.

## Example

---

```
/* REXX */

/* Process all events */
do event=1 to UFM.NUM_EVENTS
  "IDIEventInfo event"
  if RC>>0 then iterate

  :
end
```

---

*Figure 165. IDIEventInfo command example*

## IDIFREE command

The IDIFREE command can be used to unallocate (free) DDnames that might have been allocated using IDIALLOC.

### Syntax

```
►► IDIFREE DD ( ddname ) ►►
```

### Return codes

The IDIFREE command provides the following return codes:

- 0        Unallocation of all specified DDnames was successful.
- 4        Unallocation of one or more specified DDnames failed. Explanations of the errors are written to the IDITRACE DDname.
- 8        Command syntax error. An explanation of the error is written to the IDITRACE DDname.

## Example

---

```
/* REXX */

/* Free the DDnames DD1 and DD2 */
"IDIFREE DD(DD1,DD2)"
if RC = 0 then say 'Success!'
```

---

*Figure 166. IDIFREE command example*

## IDIModQry command

### IDIModQry command

The IDIModQry command is used to obtain information about a named load module from a Fault Analyzer REXX user exit. It can only be used in a Formatting user exit.

#### Syntax

```
►►—IDIMODQRY—NAME—(—module_name—)—LP—(—var_name—)——MODLEN—(—var_name—)——►►
```

where:

#### NAME(*module\_name*)

The name of the module for which the load point, and optionally module length, are being requested.

#### LP(*var\_name*)

The name of a REXX variable to hold the character format of the module load point.

#### MODLEN(*var\_name*)

The name of a REXX variable to hold the character format of the hexadecimal module length. This value is optional.

### Return codes

The IDIModQry command provides the following return codes:

- 0 Requested module found.
- 4 Requested module not found.
- 12 Syntax error.

### Example

---

```
/* REXX */  
  
"IDIMODQRY NAME(\"IDISCBL1\") LP(addr) MODLEN(len)"  
if rc = 0 then "note 'Module IDISCBL1 at address" addr "length" len'''
```

---

Figure 167. IDIModQry command example

### IDIRegisterFaultEntry command

The IDIRegisterFaultEntry command can be used to register a fault entry at any time during analysis of an MVS dump data set, for example a CICS system dump. This command allows the early creation of a fault entry in a history file, without the need to first exit interactive reanalysis. An installation can choose to automate the registration fault entry creation by issuing the IDIRegisterFaultEntry command from, for example, an Analysis Control user exit, or, alternatively, users can invoke a Formatting user exit on demand to issue this command during interactive reanalysis.

#### Note:

1. This command should not be used to register fault entries in a dump registration Analysis Control or Notification user exit, since doing so could result in extra unwanted fault entries being created. The IDIXTSEL dump registration processing automatically creates a fault entry, even if no user exit is used.
2. As an alternative to using the IDIRегистерFaultEntry command, the GenerateSavedReport option can be used to create a fault entry in the current history file for an MVS dump analyzed in batch. For details, see “GenerateSavedReport” on page 478.

### Syntax

```
►—IDIRегистерFaultEntry— [IDIHIST(—history_file_dsn—)]
```

where:

#### IDIHIST(*history\_file\_dsn*)

Specifies the history file in which the registration fault entry should be created. For interactive reanalysis, if this parameter is not specified, or if the user does not have UPDATE access to the specified history file, then a prompt is issued to allow the specification of the history file to be used.

For batch reanalysis, this parameter must be specified, otherwise, RC=4 is issued.

### Return codes

The IDIRегистерFaultEntry command provides the following return codes:

- 0** Successful completion.
- 4** A fault entry already exists, or request canceled by user via interactive prompt.
- 12** Command syntax error. An explanation of the error is written to the IDITRACE DDname.

### Example

---

```
/* REXX */

/* Create registration fault entry in history file MY.HIST */
ENV.USER_TITLE = 'My fault!'
ENV.USER_NAME  = UserID()
ENV.LOCK_FLAG  = '/'
dsn = 'my.hist'
"IDIRегистерFaultEntry IDIHIST("dsn")"
if rc <> 0 then
  "IDIWTO IDIRегистерFaultEntry failed, rc="rc
exit 0
```

---

Figure 168. IDIRегистерFaultEntry command example

See “Example 4 (Batch MVS dump registration)” on page 392 for another example of the IDIRегистерFaultEntry command usage.

## **IDIWRITE command**

The IDIWRITE command is used to pass data records from a user exit to Fault Analyzer. It can only be used in a Compiler Listing Read, Message and Abend Code Explanation, or Formatting user exit.

### **Syntax**

```
►►IDIWRITE [var_name]►►
```

where:

*var\_name*

The name of a variable containing the data record.

If the IDIWRITE command is used without *var\_name*, data records must be passed using the exit-specific data area as described for the relevant exit types.

### **Return codes**

The IDIWRITE command provides the following return codes:

- 0      The record was written successfully.
- 2      Writing of records has been disabled due to previous errors.
- 4      The record was not written due to errors. An explanation of the error is written to the IDITRACE DDname.
- 8      Command syntax error. An explanation of the error is written to the IDITRACE DDname.

## Example

---

```
/* REXX */
/* Pass records to Fault Analyzer */

/* Method 1 - plain text */
"IDIWRITE 'This is record 1'"
if RC = 0 then say 'Method 1 success!'

/* Method 2 - using LST data area (Compiler Listing Read user exit) */
rec = 'This is record 2'
LST.DATA_LENGTH = length(rec)
LST.DATA_BUFFER = rec
"IDIWRITE"
if RC = 0 then say 'Method 2 success!'

/* Method 3 - letting REXX resolve data record variable */
rec = 'This is record 3'
"IDIWRITE '"rec"'"
if RC = 0 then say 'Method 3 success!'

/* Method 4 - letting Fault Analyzer resolve data record variable */
rec = 'This is record 4'
"IDIWRITE rec"
if RC = 0 then say 'Method 4 success!'
```

---

Figure 169. IDIWRITE command example

## IDIWTO command

The IDIWTO command is used to write a message to the MVS console. This command can, for example, be used instead of the REXX SAY command whenever tracing is not active (IDITRACE DDname not allocated).

### Syntax

►—IDIWTO—message\_text—►

New-line characters (X'15') in the message text can be used to split long messages into multiple WTOs. Any other non-printable characters are changed to periods.

### Return codes

The IDIWTO command always completes with RC=0.

## Example

---

```
/* REXX */

/* Write a message to the MVS console */
"IDIWTO Minutes since last duplicate fault =" EPC.MINUTES_SINCE_LAST_DUP
```

---

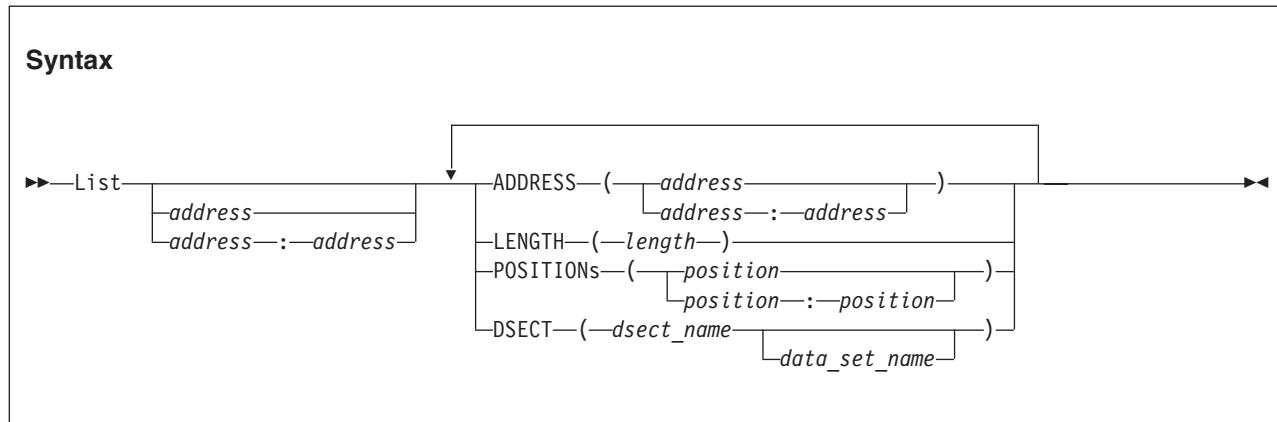
Figure 170. IDIWTO command example

## IDIWTO command

To avoid possible confusion, it is recommended that any WTO messages that are issued do not include a message ID that could be mistaken for one of the formal messages issued by Fault Analyzer. If you want to prefix message IDs with "IDI", then you might add an extra character to eliminate the chance of matching a Fault Analyzer message—for example, "IDIX" followed by a number.

## List command

The List command can be used by a Formatting user exit to print storage areas from the analyzed fault environment.



## Parameters

### address

### start:end

Specifies either the start address, or an address range, as a positional parameter. This address is an alternative to using the ADDRESS keyword. See ADDRESS below for details about valid syntax.

### ADDRESS(address)

### ADDRESS(address:address)

Specifies the address of the storage to be printed. Specify either a single address or as an address range.

The address parameter is 64-bit enabled. For details, see "Specifying 64-bit addresses" on page 93. The address may optionally follow by a period (for example, "000176C0.".).

### LENGTH(length)

Specifies the number of bytes to be printed. Specify as a hexadecimal value (indicated by X'...') or as a decimal value (no indication required). If an address range is specified, then specification of LENGTH is ignored.

### POSITIONs(position)

### POSITIONs(position:position)

Specifies the offset from the start address to the first byte of storage to be printed. Specify as either a single offset or as an offset range. Both POSITION and POSITIONS are accepted.

All offsets must be either hexadecimal (indicated by X'...') or signed decimal (optionally indicated by F'...').

**DSECT(dsect\_name)****DSECT(dsect\_name data\_set\_name)**

Specifies the name of a DSECT mapping member to be used when formatting storage at the requested address.

If a data set name is not specified, then the DSECT must be available through the IDIDSECT DDname (for details, see “DataSets” on page 457). Otherwise, the DSECT must exist in the specified data set name.

If a parameter is specified multiple times, then only the last specification takes effect.

If the DSECT parameter is not specified, then storage is formatted with either 16 or 32 bytes per line (depending on preferred formatting width specification for the type of fault analysis being performed), showing both hexadecimal and EBCDIC values.

If the DSECT parameter is specified, then storage is shown formatted the same as if the DSECT command was used. For details, see “Mapping storage areas using DSECT information” on page 154.

### Return codes

The List command provides the following return codes:

- 0** Command completed successfully.
- 4** An error occurred during DSECT formatting. An explanation of the error is written to the IDITRACE DDname.
- 12** Command syntax error. An explanation of the error is written to the IDITRACE DDname.

### Example

```
/* REXX */
"LIST 0 LENGTH(128)"
```

Figure 171. List command example

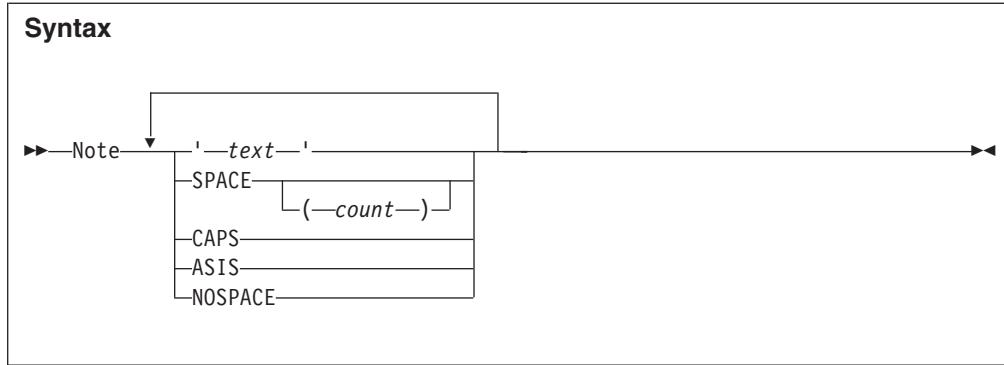
The above example might produce the following output:

Address	Offset	Hex	EBCDIC
00000000		040C0000 810692C8 00000000 00000000	*.....a.kH.....*
00000010	+10	00FC7F08 00000000 070C1000 85532492	*.."......e..k*
00000020	+20	078D0000 00FC7F5A 078D1000 8775BB42	*....."!....g...*
00000030	+30	00000000 00000000 070C0000 85532496	*.....e..o*
00000040	+40	00000000 00000000 00000000 00FC7F08	*.....".*
00000050	+50	00000000 00000000 040C0000 810676D0	*.....a..}*
00000060	+60	040C0000 80FFB080 00080000 BF286880	*.....*
00000070	+70	00080000 BF287940 040C0000 81068A00	*.....`....a...*

### Note command

The Note command can be used to by a Formatting user exit to print a line of text.

## Note command



### Parameters

**'text'** Specifies the text to be printed, enclosed in apostrophes. Either single quotes ('') or double quotes ("") are permitted, as long as the same type is used as both the first and the last character.

Any quotes within the text, of the same type as used to enclose the string, must be specified twice. For example, to print

It isn't there

specify the text as either

"It isn't there"

or

'It isn''t there'

If no text is specified, then no data is printed. However, any blank lines specified using the SPACE parameter are still written.

#### SPACE

##### SPACE(count)

Specifies the number of blank lines to be written ahead of the next text line.

If the SPACE parameter is specified without *count*, then it defaults to 1.

**CAPS** Specifies that text is written in all uppercase characters.

**ASIS** Specifies that text is written "as is" without any uppercasing being performed.

#### NOSPACE

Specifies that no blank lines are written ahead of the next text line. This value is the default.

If a parameter is specified multiple times, then only the last specification takes effect.

### Return codes

The Note command provides the following return codes:

**0** Command completed successfully.

**12** Command syntax error. An explanation of the error is written to the IDITRACE DDname.

## Example

---

```
/* REXX */

"NOTE 'This is a simple note.'"
"NOTE 'This note follows the previous without any blank lines inserted.'"
"NOTE 'This note has 2 blank lines ahead of it.' SPACE(2)"
```

---

*Figure 172. Note command example*

The above example produces the following output:

This is a simple note.  
This note follows the previous without any blank lines inserted.

This note has 2 blank lines ahead of it.

---

## Formatting tags

The following describes the tags that are available to a Formatting user exit when formatting data for the report. The tags provide a way to create headings, lists, and so on for the displayed data using HTML-like syntax. The tag stream is passed back to Fault Analyzer from the Formatting user exit using the IDIWrite command.

An example showing the use of the formatting tags follows:

## Formatting tags

---

```
"IDIWRITE '<P>First paragraph.'"
"IDIWRITE '<AREA INDENT=5>'"
"IDIWRITE '<P>Second paragraph, indented 5 characters from the first. ''"
"IDIWRITE 'This <DATA 3><P> tag is treated as text only.'"
"IDIWRITE '<P COMPACT>Third paragraph. ''"
"IDIWRITE 'Note that this paragraph is not preceded by a blank line.'"
"IDIWRITE '</AREA>'"
"IDIWRITE '<P>Fourth paragraph - now we are back at the left margin.</P>'"
"IDIWRITE '<L>***** This line will '''
"IDIWRITE '<HP>not</HP> wrap at the preferred formatting width!''"
"IDIWRITE '<P><ADDR 625f22>Previous area</ADDR> and <ADDR 625f22></ADDR> are '''
"IDIWRITE 'both point-and-shoot fields to the Dump Storage ''"
"IDIWRITE 'display for address 00625F22 in the interactive reanalysis report.'"
"IDIWRITE '<DL BREAK=STDLBL>'"
"IDIWRITE '<DT>This is a long definition term''"
"IDIWRITE '<DD>This is the matching definition description which might wrap '''
"IDIWRITE 'depending on the preferred formatting width.'"
"IDIWRITE '<DT>A shorter definition term''"
"IDIWRITE '<DD>The definition description of the second term.'"
"IDIWRITE '</DL>'"
"IDIWRITE '<P><DUMP 0 20>Address 0 storage for a length of 32 bytes:</DUMP>'"
"IDIWRITE '<UL>'"
"IDIWRITE '<LI>In an unordered list, each item is preceded by a bullet. ''"
"IDIWRITE 'If necessary, the item description will wrap at the ''"
"IDIWRITE 'preferred formatting width.'"
"IDIWRITE '<LI>Another item in the same list.'"
"IDIWRITE '</UL>'"
"IDIWRITE '<P><NOTEL>'"
"IDIWRITE '<LI>In a note list, each note is numbered and the list is '''
"IDIWRITE 'preceded by a ""Notes:"" heading. If necessary, the note '''
"IDIWRITE 'description will wrap at the preferred formatting width.'"
"IDIWRITE '<LI>Another note in the same list.'"
"IDIWRITE '</NOTEL>'"
"IDIWRITE '<P><TH>Column Column</TH>'"
"IDIWRITE '<L><U>1      <U>2      </U>'"
"IDIWRITE '<L>    123      17'"

exit 0
```

---

*Figure 173. Sample REXX Formatting user exit 3 source*

Formatted, the above might appear as follows (point-and-shoot fields and highlighted text shown in bold style):

```

File View Services Help
Interactive Reanalysis Options Line 1 Col 1 80
Command ===> Scroll ===> CSR
JOBNAME: P35777 SYSTEM ABEND: 0C7 FAE1 2008/01/31 22:51:13

First paragraph.

Second paragraph, indented 5 characters from the first. This <P> tag is
treated as text only.
Third paragraph. Note that this paragraph is not preceded by a blank
line.

Fourth paragraph - now we are back at the left margin.
***** This line will not wrap at the prefer

Previous area and 00625F22 are both point-and-shoot fields to the Dump Storage
display for address 00625F22 in the interactive reanalysis report.

This is a long definition
term. . . . . : This is the matching definition description
which might wrap depending on the preferred
formatting width.

A shorter definition term . : The definition description of the second term.

Address 0 storage for a length of 32 bytes:
Address Offset Hex EBCDIC / ASCII
00000000 040C0000 810692C8 00000000 00000000 *....a.kH.....*
00000010 +10 00FC7F08 00000000 070E0000 00000000 *..".....*

o In an unordered list, each item is preceded by a bullet. If necessary,
the item description will wrap at the preferred formatting width.

o Another item in the same list.

Notes:

1. In a note list, each note is numbered and the list is preceded by a
"Notes:" heading. If necessary, the note description will wrap at the
preferred formatting width.

2. Another note in the same list.

Column Column
1 2
123 17

```

Figure 174. Sample REXX Formatting user exit 3 output

The above example is provided in softcopy format as member IDISUFM3 in data set IDI.SIDISAM1.

General rules for the formatting tags:

- All blanks are significant, except at the beginning and end of lines in a paragraph, and at the beginning and end of definition descriptions (text preceded by the <DD> tag).
- Text, including blank characters, that is not preceded by any tag implicitly causes a <P> tag to be inserted ahead of the text.

## Formatting tags

- All tags and attributes are non-case-sensitive.
- The maximum line width of any output is 132 characters. Beyond this, the text wraps.

The following explains each tag in detail.

### ADDR (address)

The ADDR tag defines an address point-and-shoot field.

When the address field is displayed in an interactive reanalysis report, the user can place the cursor on the field and press Enter to invoke the Dump Storage display at the specified address.

#### Syntax

```
►--<ADDR--address--> [ hotspot-text ] [ </ADDR> ] --►
```

#### address

The hexadecimal address to be displayed when selecting the point-and-shoot field.

The address parameter is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

#### hotspot-text

The point-and-shoot field text to be displayed. If not specified, then the address is used.

### Description

The ADDR tag does not cause a line break.

In the interactive reanalysis report, the point-and-shoot field text is presented in yellow.

### AREA (area)

The AREA tag defines a section of a display. It is used to control indentation of text beyond what the default indentation provided for the higher level tags provide.

#### Syntax

```
►--<AREA--> [ area-text ] [ </AREA> ] --►
```

INDENT=  
  10  
  n

#### INDENT

This attribute specifies the number of characters by which the current indentation is incremented.

**area-text**

These are tags and text for the display section.

**Description**

The AREA tag can be nested within another AREA tag. Any other tags are implicitly ended.

**DD (definition description)**

The DD tag defines the description of a term in a definition list.

**Syntax**

```
►—<DD>—description-text—►
          |—————</DD>—————►
```

**description-text**

This value is the term description.

**Description**

See "DL (definition list)."

**DATA (data)**

The DATA tag defines the number of characters that follow the tag for which no tag processing is to be performed. This tag is generally only used if the text to be written contains characters that might otherwise be misinterpreted as formatting tags.

**Syntax**

```
►—<DATA—length>—►
```

**length** The number of characters in the input stream immediately following the DATA tag that should be treated as textual data regardless of any possible tag contents.

**Description**

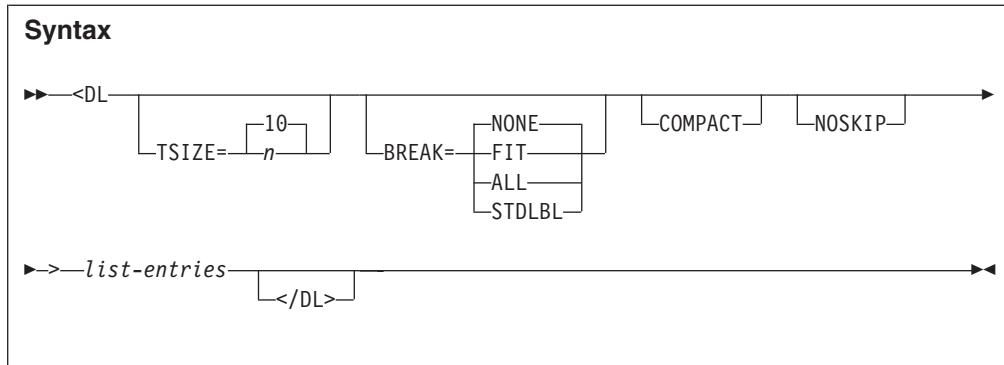
Text that might contain valid tags can be protected from causing formatting problems by using the DATA tag to specify the number of characters that should be treated as text only.

**Note:** The use of tag delimiters ('<' or '>') in text that is not preceded by a DATA tag is permitted as long as no valid tag syntax occurs. Invalid and incomplete tags are returned to the input stream as textual data.

**DL (definition list)**

The DL tag defines a list of terms and their corresponding definitions.

## DL (definition list)



**TSIZE** This attribute specifies the indentation of the definition description. The minimum value is 3 characters and the default value is 10 characters.

### BREAK

This attribute controls the formatting of the definition terms and descriptions:

- If BREAK=NONE, then the term is on the same line as the description, spilling into the description area if the length exceeds TSIZE. This value is the default.
- If BREAK=FIT, then the description is on the line below the term if the term exceeds the TSIZE value.
- If BREAK=ALL, then every definition is on the line below the term.
- If BREAK=STDLBL, then a standard Fault Analyzer label is used. If the term length exceeds the TSIZE value in effect, then the text is wrapped within the TSIZE width. Trailing periods and a colon are added to the last term line.

### COMPACT

This attribute causes the list to format without a blank line between the items in the list.

### NOSKIP

This attribute causes the list to format without creating a blank line before the first line of the list.

### list-entries

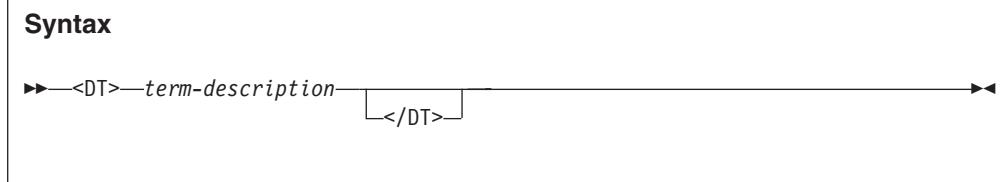
These are the DT and DD tags that make up the list entries.

### Description

In the interactive reanalysis report, the list entries are presented in white. However, if the STDLBL attribute is used, then the definition terms are in green.

## DT (definition term)

The DT tag defines a term in a definition list.



**term-description**

This value is specified via the DD tag.

**Description**

See “DL (definition list)” on page 429.

**DUMP (EBCDIC dump)**

The DUMP tag inserts a hex-dump display in-line in the formatted output, as opposed to the ADDR tag which only shows the address itself. This insertion is particularly useful when formatting for the batch report, since the user is not otherwise able to view the storage.

**Syntax**

```
►--<DUMP--address--length--> [heading-text] [</DUMP>] --►
```

**address**

Hexadecimal address of storage area to be displayed.

The address parameter is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

**length** Hexadecimal length (in bytes) of storage area to be displayed.

**heading-text**

The heading text to immediately precede the hex-dump display.

**Description**

The hex-dump display always starts in column one, regardless of current indentation.

The character-interpreted section on the right-hand side of the display is based on EBCDIC-encoded hexadecimal values. If the data is known to contain ASCII-encoded values, then use the DUMPA tag instead (see “DUMPA (ASCII dump)”).

In the interactive reanalysis report, the heading text is presented in white.

**DUMPA (ASCII dump)**

The DUMPA tag inserts a hex-dump display in-line in the formatted output, as opposed to the ADDR tag which only shows the address itself. This insertion is particularly useful when formatting for the batch report, since the user is not otherwise able to view the storage.

**Syntax**

```
►--<DUMPA--address--length--> [heading-text] [</DUMPA>] --►
```

## DUMPA (ASCII dump)

### address

Hexadecimal address of storage area to be displayed.

The address parameter is 64-bit enabled. For details, see “Specifying 64-bit addresses” on page 93.

**length** Hexadecimal length (in bytes) of storage area to be displayed.

### heading-text

The heading text to immediately precede the hex-dump display.

### Description

The hex-dump display always starts in column one, regardless of current indentation.

The character-interpreted section on the right-hand side of the display is based on ASCII-encoded hexadecimal values. If the data is known to contain EBCDIC-encoded values, then use the DUMP tag instead (see “DUMP (EBCDIC dump)” on page 431).

In the interactive reanalysis report, the heading text is presented in white.

## HP (highlighted phrase)

The HP tag identifies text to be displayed with highlighted emphasis.

**Note:** Highlighting is only available in the interactive reanalysis report.

### Syntax

```
►—<HP>—text-to-be-highlighted—►  
                                        └─</HP>─┘
```

### text-to-be-highlighted

This text displays with highlighted emphasis.

### Description

This tag does not cause a line break.

In the interactive reanalysis report, the text is presented in turquoise.

## L (line)

The L tag defines a line of text that is not subject to the user preferred display width.

### Syntax

```
►—<L>—[line-text]—</L>—►
```

### line-text

The line text.

**Description**

Each line formats as an unindented non-flowing line of text. No blank lines are added before or after the line text.

The width of the line is determined by the maximum line width of the display. No line should exceed this limit as it then flows onto the following line.

In the interactive reanalysis report, the line text is presented in white.

**LI (list item)**

The LI tag defines a list item within a note list or an unordered list.

**Syntax**

```
►-<LI>—item-text—</LI>
```

**item-text**

The item text.

**Description**

In the interactive reanalysis report, the line text is presented in white.

**NOTEL (note list)**

The NOTEL tag defines a list of notes.

**Syntax**

```
►-<NOTEL>—  
    [COMPACT] [NOSKIP]—>list-entries—</NOTEL>
```

**COMPACT**

This attribute causes the list to be formatted without a blank line between the list items.

**NOSKIP**

This attribute causes the list to format without creating a blank line before the first line of the list.

**list-entries**

These are specified using the LI tag.

**Description**

This tag causes a line break.

**P (paragraph)**

The P tag defines a paragraph of text.

## P (paragraph)

### Syntax

```
►-<P> [COMPACT] > [paragraph-text] [</P>]-►
```

### COMPACT

This attribute causes the paragraph to format without a blank line before the paragraph.

### paragraph-text

The text of the paragraph.

### Description

Each paragraph formats as an unindented flowing block of text. A blank line is added before the paragraph unless the COMPACT attribute is specified.

The width of the displayed paragraph is determined by the preferred formatting width specified by the user via the View pull-down menu.

Paragraphs within a list align with the text of the list item.

In the interactive reanalysis report, the paragraph text is presented in white.

## TH (table heading)

The TH tag defines a table heading line.

### Syntax

```
►-<TH> [heading-text] [</TH>]-►
```

### heading-text

The table heading line.

### Description

Each table heading line formats as an unindented non-flowing line of text. No blank lines are added before or after the line text.

The width of the line is determined by the maximum line width of the display. No line should exceed this limit as it then flows onto the following line.

In the interactive reanalysis report, the table heading line text is presented in blue.

## U (underline)

The U tag identifies text to be displayed underlined.

**Note:** Underlining is only available in the interactive reanalysis report.

**Syntax**

```
►—<U>—text-to-be-underlined—</U>—►
```

**text-to-be-underlined**

This text displays underlined.

**Description**

This tag does not cause a line break.

In the interactive reanalysis report, the underlined text is presented in blue.

**UL (unordered list)**

The UL tag defines an unordered list.

**Syntax**

```
►—<UL—[COMPACT]—[NOSKIP]—>—list-entries—</UL>—►
```

**COMPACT**

This attribute causes the list to be formatted without a blank line between the list items.

**NOSKIP**

This attribute causes the list to format without creating a blank line before the first line of the list.

**list-entries**

These are specified using the LI tag.

**Description**

This tag causes a line break.

**The IDIXUFMT load module Formatting user exit**

The IDIXUFMT load module Formatting user exit (in the following topics referred to as the IDIXUFMT exit) is a special type of user exit. It does not follow the normal rules that apply to other user exits described in this chapter, whether in REXX or in load module format.

The IDIXUFMT exit cannot be specified through the Exits option, but is located by load module name. If a load module with the name IDIXUFMT is found in an APF-authorized library during Fault Analyzer execution, then it is invoked during the formatting of the analysis report. The IDIXUFMT exit is called immediately before any other Formatting user exits specified through the Exits option.

The IDIXUFMT exit:

- Must be LE-compliant.
- Must not contain a "main" function.

## The IDIXUFMT load module Formatting user exit

- Must be link-edited with the NORENT option.

Fault Analyzer invokes the IDIXUFMT exit by way of the IDIXMFMT entry point, which is contained within IDIXLFMT, the non-executable load module that is provided with Fault Analyzer. The exit user code is invoked by way of entry point IDIXUFMT.

## Entry specifications

The user code IDIXUFMT entry point is invoked with:

- R1 pointing to two fullwords:
  - The first fullword is the address of the ENV data area (see “ENV - Common exit environment information” on page 513).
  - The second fullword is the address of the UFM data area (see “UFM - Formatting user exit parameter list” on page 525).
- R13 pointing to save area.
- R14 containing the return address.
- R15 containing the IDIXUFMT entry point address.

## Return specifications

On return from the IDIXUFMT entry point:

- R0 and R1 are undefined.
- R2 through R14 must be unchanged.
- R15 is undefined.

## Sample IDIXUFMT exits

Two sample IDIXUFMT exits are provided, along with JCL to compile (or Assemble) and link:

- A sample C IDIXUFMT exit is provided in IDI.SIDISAM1(IDIXUFMC)
- A sample Assembler IDIXUFMT exit is provided in IDI.SIDISAM1(IDIXUFMA)

## IDIXUFMT functions

There are many Fault Analyzer functions available for use from within the IDIXUFMT exit.

All functions have uppercase names, consisting of 8 characters, or less. These basic names permit an exit that is written in, for example High Level Assembler, to call these functions.

All functions use C linkage.

The following topics describe these functions.

### IDIXDLOC – Locate dump storage

#### Format:

```
#include "idixufmh.h"  
  
int IDIXDLOC(int addr, int len);
```

**General description:** The IDIXDLOC() function is used to access storage in the analyzed address space by virtual address and length.

For real-time processing, the storage that is accessed is generally that of the actual address space which is being analyzed. For reanalysis, the storage is obtained from the mini-dump.

Regardless of real time or reanalysis mode, do not dereference storage areas that are not obtained by calling the IDIXDLOC() function, since protection exceptions might otherwise occur.

IDIXDLOC() is functionally equivalent to the Fault Analyzer REXX command "Evaluate".

**Returned value:** If the storage is available for the requested length, then IDIXDLOC() returns the address of the storage area.

IDIXDLOC() returns a negative value if the requested address is available, but for a length that is less than the requested length. The available length can be determined by subtracting the returned value from 0. To obtain the address of the partial storage area, call IDIXDLOC() again with the reduced length.

A zero value is returned if the requested address is not available, regardless of length.

**Example:**

```
#include "idixufmh.h"

int maddr, cvt;
maddr = IDIXDLOC(16,4); /* Get address of CVT pointer /
if (maddr > 0)
    cvt = (int )maddr; /* Get CVT pointer */
```

## **IDIXEINF – Obtain event information**

**Format:**

```
#include "idixhfmt.h"

int IDIXEINF(UFM *p_ufm, int event_no);
```

**General description:** The specified UFM data area is populated with information applicable to the specified event number.

IDIXEINF() is functionally equivalent to the Fault Analyzer REXX command "IDIEventInfo".

**Returned value:** IDIXEINF() returns zero if the information was retrieved successfully.

IDIXEINF() returns non-zero if no information is available for the specified event number. An explanation of the error is written to the IDITRACE DDname.

**Example:**

```
#include "idixhfmt.h"

UFM ufm;
int rc;
rc = IDIXEINF(&ufm, 1);
If (!rc) { // Successful completion
    ...
}
```

## IDIXGETN - Get data area decimal character field value

### IDIXGETN – Get data area decimal character field value

#### Format:

```
#include "dixhfmt.h"

#define IDIXGETN(pSrc) \
    (IDIXGETN)(pSrc, sizeof(pSrc)) \
int     (IDIXGETN)(char *pSrc, int src_len);
```

**General description:** IDIXGETN() is used to convert a data area character field that contains decimal characters to an "int" value.

IDIXGETN() is a macro, which calls a function by the same name. The advantage of using the macro is that only a single argument is required.

**Returned value:** If successful, IDIXGETN() returns the converted signed "int" value, represented in the string. If unsuccessful, it returns an undefined value.

#### Example:

```
#include "idixhfmt.h"

int num_events;
UFM ufm;
...
num_events = IDIXGETN(ufm.NUM_EVENTS);
```

## IDIXGETS - Get data area character field as C string

#### Format:

```
#include "idixhfmt.h"

#define IDIXGETS(pSrc) \
    (IDIXGETS)(pSrc, sizeof(pSrc)) \
char * (IDIXGETS)(char *pSrc, int src_len);
```

**General description:** IDIXGETS() is used to create a NULL-terminated string from a data area character field. For data area fields not in buffered data format, the string is created in the LE HEAP storage and is automatically free'd when the IDIXUFMT exit processing finishes. The caller of IDIXGETS() must not free the storage area returned. For data area fields in buffered data format, the existing buffer address is returned.

It is not possible to update a data area field by modifying the string that is returned by IDIXGETS().

IDIXGETS() is a macro, which calls a function by the same name. The advantage of using the macro is that only a single argument is required.

**Returned value:** Returns the address of the requested data area field as a NULL-terminated string.

#### Example:

```
#include "idixhfmt.h"

UFM ufm;
char *psz;
...
...
```

```

psz = IDIXGETS(ufm.EVENT_TYPE);
If (strlen(psz) >= 6 && memcmp(psz,"Abend ") == 0) { // Abend event
    ...
}

```

### IDIXGETX – Get data area hexadecimal character field value

**Format:**

```

#include "idixhfmt.h"

#define IDIXGETX(pSrc) \
    (IDIXGETX)(pSrc, sizeof(pSrc)) \
int     (IDIXGETX)(char *pSrc, int src_len);

```

**General description:** IDIXGETX() is used to convert a data area character field that holds hexadecimal characters (0-9 or A-F) to an "int" value.

IDIXGETX() is a macro, which calls a function by the same name. The advantage of using the macro is that only a single argument is required.

**Returned value:** If successful, IDIXGETX() returns the converted signed "int" value, represented in the string. If unsuccessful, it returns an undefined value.

**Example:**

```

#include "idixhfmt.h"

int pgm_len;
UFM ufm;
...
pgm_len = IDIXGETX(ufm.PROGRAM_LENGTH);

```

### IDIXLIST – Print storage area in report

**Format:**

```

#include "idixhfmt.h"

int IDIXLIST(int addr, int len);

```

**General description:** IDIXLIST() can be used to print storage areas from the analyzed fault environment. The addr argument must be a virtual dump address.

The format of the printed storage area includes a heading, and both hexadecimal and EBCDIC/ASCII representation of the storage, as shown in the following example:

Address	Offset	Hex	EBCDIC / ASCII
00000000		040C0000 810692C8 00000000 00000000	*.....a.kH.....*
00000010	+10	00FC7F08 00000000 070C1000 85532492	*..".....e..k*
00000020	+20	078D0000 00FC7F5A 078D1000 8775BB42	*....."!....g...*
00000030	+30	00000000 00000000 070C0000 85532496	*.....e..o*
00000040	+40	00000000 00000000 00000000 00FC7F08	*.....".*
00000050	+50	00000000 00000000 040C0000 810676D0	*.....a..}*
00000060	+60	040C0000 80FFB080 00080000 BF286880	*.....*
00000070	+70	00080000 BF287940 040C0000 81068A00	*.....` ....a...*

IDIXLIST() is functionally equivalent to the Fault Analyzer REXX command "List".

**Returned value:** IDIXLIST() always returns zero.

## IDIXLIST - Print storage area in report

### Example:

```
#include "idixhfmt.h"

IDIXLIST(0,128); // Show address 0 for a length of 128 bytes
```

## IDIXNOTE – Write simple line of text to report

### Format:

```
#include "idixhfmt.h"

int IDIXNOTE(char *psz, ...);
```

**General description:** IDIXNOTE() writes a line of unformatted text in the report. The psz argument must point to a NULL-terminated string. (To write formatted text, use the IDIXWRIT() function instead.)

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "Note".

**Returned value:** IDIXNOTE() always returns zero.

### Example:

```
#include "idixhfmt.h"

int i;

IDIXNOTE("Important data follows");
...
IDIXNOTE("A total of %d entries listed.",i);
```

## IDIXTRCE – Write simple line of text to IDITRACE

### Format:

```
#include "idixhfmt.h"

int IDIXTRCE(char *psz, ...);
```

**General description:** IDIXTRCE() can be used to perform the following functions:

- Control the tracing of IDI\* function calls by writing information to the IDITRACE DDname about passed parameters and final return code.
  - If the psz arg is specified as "(char )-1", then IDI\* function call tracing is enabled.
  - If the psz arg is specified as "NULL", then IDI\* function call tracing is disabled
- Write a line of unformatted text to the IDITRACE DDname. The psz argument must point to a NULL-terminated string. This mode of the function is equivalent to using the REXX command "SAY" from within a Fault Analyzer REXX user exit.

If the psz argument is a format string suitable for use by the C sprintf() function, then extra required arguments can follow.

**Returned value:** IDIXTRCE() always returns zero.

**Example:**

```
#include "idixhfmt.h"

int i = 5;
IDITRACE((char *)-1); /* start IDI* function tracing */
IDIXTRCE("Couldn't format data area ABC.");
IDIXTRCE("A total of %d control blocks formatted.", i);
```

**IDIXWRIT – Write formatted text to report****Format:**

```
#include "idixhfmt.h"

int IDIXWRIT(char *psz, ...);
```

**General description:** The IDIXNOTE() function can be used only to write a simple line of text to the report. However, the string that is passed to the IDIXWRIT() function in the psz argument can contain formatting tags (see “Formatting tags” on page 425).

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "IDIWRITE".

**Returned value:** IDIXWRIT() always returns zero.

**Example:**

```
#include "idixhfmt.h"

int i;
IDIXWRIT("<DL>");
for (i = 0; i < 10; ++i) {
    IDIXWRIT("<DT>Item %d</DT>", i);
    IDIXWRIT("<DD>Item description</DD>");
}
IDIXWRIT("</DL>");
```

**IDIXWTO – Write message to MVS console****Format:**

```
#include "idixhfmt.h"

int IDIXWTO(char *psz, ...);
```

**General description:** IDIXWTO() is used to write a message to the MVS console. The psz argument must point to a NULL-terminated string.

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "IDIWTO".

**Returned value:** IDIXWTO() always returns zero.

**Example:**

```
#include "idixhfmt.h"

IDIXWTO("Unable to complete IDIXUFMT exit processing!");
```

## **IDIXWTO - Write message to MVS console**

---

## Chapter 32. Installing non-ISPF interfaces to access Fault Analyzer history files

The following describes installation requirements for optional interfaces which allow access to Fault Analyzer history files from platforms other than TSO/ISPF.

---

### Installing the Fault Analyzer plug-in for Eclipse

This feature is an optional feature.

Refer to “Using the Fault Analyzer plug-in for Eclipse” on page 205 for information about using this interface.

**Note:** It is a requirement for this feature that the IBM Problem Determination Tools for z/OS Common Component (PDTCC) is installed.

To install the CICS Explorer plug-in, the following steps must be performed:

1. On the host MVS system:

- a. Customize the IBM Problem Determination Tools for z/OS Common Component Common Server

**Note:** The Fault Analyzer server for the Eclipse plug-in is required to be installed, regardless of whether IBM Rational Developer for System z is also installed. Although both environments can co-exist, they require separate servers.

2. On the client PC:

- a. Download and install CICS Explorer
- b. Download Fault Analyzer plug-in
- c. Install the Fault Analyzer plug-in into CICS Explorer

**Note:** The Fault Analyzer client for IBM Rational Developer for System z cannot be installed into CICS Explorer—only the Fault Analyzer plug-in for Eclipse can be used in this environment.

These steps are described in more detail in the following sections.

### Customize the IBM Problem Determination Tools for z/OS Common Component Common Server

Refer to *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* for information about installing the IBM Problem Determination Tools for z/OS Common Component Common Server.

A sample Fault Analyzer configuration file for the IBM Problem Determination Tools for z/OS Common Component Common Server is provided as member IDIGSVRJ in data set IDI.SIDISAM1:

```
CONFIG=FA
SPAWN_PROGRAM=IDIGMAIN
SPAWN_REGIONSZ=500 1
```

## Installing the Fault Analyzer plug-in for Eclipse

```
SPAWN_JOBNAME=IDISVRF  
SPAWN_PARMS_SECTION  
ISPF_PROF_DSN=&USERID..ISPF.ISPPROF 2  
ISPF_APPL=IDI 3
```

The parameters you might need to change are as follows:

**1** Increase SPAWN\_REGIONSZ if downloading extremely large fault entry minidumps with more than 80,000 pages.

**2** Change &USERID..ISPF.ISPPROF if a different data set is used for ISPF profile members.

&USERID is replaced by the user ID of the connected user. For example, if the user ID is FRED, and the ISPF application ID in **3** is IDI, then the ISPF profile is retrieved from FRED.ISPF.ISPPROF(IDIPROF).

To use only a part of the user ID as the substitution value, substring specification can be used (for details, see “Symbol substring specification” on page 462). For example, to use only the first 3 characters of the user ID as the high-level qualifier, specify &USERID(1:3)..ISPF.ISPPROF.

**3** Change IDI if a different ISPF application ID is used for Fault Analyzer. The default value is IDI.

## Download and install CICS Explorer

The Fault Analyzer plug-in is intended to be used as a plug-in to CICS Explorer, or alternative Eclipse environment, on your PC.

CICS Explorer can be downloaded via the following web site:  
<http://www-01.ibm.com/software/awdtools/deployment/pdtplugins/>

Go to this site and follow the instructions for downloading and installing CICS Explorer.

## Download Fault Analyzer plug-in

The Fault Analyzer plug-in for Eclipse can be downloaded to your PC from the IBM Problem Determination Tools Plug-ins web site:

<http://www-01.ibm.com/support/docview.wss?uid=swg24033351>

## Install Fault Analyzer plug-in into CICS Explorer

Having downloaded the zip file, extract the contents of the compressed file to any writable directory of your choice, for example C:\FaultAnalyzer

Once the files have been extracted, double-click the file that is called readme.html, which provides installation instructions for the plug-in.

---

## Installing the Fault Analyzer client for IBM Rational Developer for System z

This feature is an optional feature.

Refer to “Using the Fault Analyzer client for IBM Rational Developer for System z” on page 205 for information about using this interface.

The Fault Analyzer feature for the host is included as part of the IBM Rational Developer for System z Version 7.1 and later IBM SMP/E for z/OS installation. In

## Installing the Fault Analyzer client for IBM Rational Developer for System z

addition to the IBM Rational Developer for System z host configuration, Fault Analyzer V7.1 with PTF UK25564 or later must be configured in your environment.

Refer to <http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp?topic=/com.ibm.etools.fa.FAClientHelpPlugin/features.html> for more information about installing this interface.

**Note:**

1. The Fault Analyzer server for the Eclipse plug-in is not used by the Fault Analyzer client for IBM Rational Developer for System z. Although both environments can co-exist, they require separate servers.
2. The Fault Analyzer Eclipse plug-in cannot be installed into IBM Rational Developer for System z—only the Fault Analyzer client for RDz can be used in this environment.

---

## Enabling interactive reanalysis under CICS

This feature is an optional feature. It allows the use of the Fault Analyzer ISPF interface to view history files and fault entries from a CICS logon, without the need for a TSO logon.

Refer to “Performing interactive reanalysis under CICS” on page 206 for information about using this interface.

**Note:** It is a requirement for this feature that the IBM Problem Determination Tools for z/OS Common Component (PDTCC) has been installed.

It is recommended that a special CICS region be set up for running Fault Analyzer in this mode to control the CPU and storage usage, without affecting other normal CICS transactions. The reason is that in this mode, the Fault Analyzer analysis programs are running in attached subtasks in the CICS region, and the JCL REGION= size needs to be large enough to accommodate the number of expected users concurrently logged on to run Fault Analyzer. The amount of storage needed varies depending on the size and complexity of the fault being analyzed, but a starting point of 32 megabytes per concurrent user is recommended. This storage is not CICS DSA storage, and needs to be available to satisfy the MVS GETMAIN requests in the attached subtasks. That is, the estimated storage requirement (for example, 10x32M=320M) needs to be at least the difference between the REGION= size and the CICS EDSALIM value.

The other setup requirements for a CICS region to run Fault Analyzer as an interactive CICS transaction are:

1. Make the required CICS resource definitions.
2. Make the required CICS JCL changes.

These requirements are described in the following sections.

## Making the required CICS resource definitions

A sample job has been provided as member IDIWCIDI in the IDI.SIDISAM1 data set, that can be used to make the required CICS resource definitions. There is one transaction definition and one associated program definition required. Optionally, there is also a transaction profile definition, which specifies the SCRNSIZE(ALTERNATE) option so that different screen sizes can be used.

### Making the required CICS JCL changes

The following modifications need to be made to your CICS JCL. It is assumed that IDI is the data set name high-level qualifier that was used during Fault Analyzer installation.

1. Add data set IDISIDIAUTH to the DFHRPL concatenation of the CICS JCL.
2. Allocate a new profile data set to be assigned to the IPVPROF and IPVTLIB DD names below—for example, IDI.IDIPPROF. This data set should be defined as a PDS(E) with LRECL=80 and RECFM=FB. Only a small data set is required, for example 5 tracks primary and 5 tracks secondary space allocation.

**Note:** Each user has their ISPF-like settings written to the IPVPROF data set, and as such each user needs UPDATE access to this data set.

3. Add the following DD names to the CICS JCL:

```
//IPVPLIB  DD DISP=SHR,DSN=IPV.SIPVPENU19
//          DD DISP=SHR,DSN=IPV.SIPVMENU19
//          DD DISP=SHR,DSN=IDI.SIDIPLIB
//          DD DISP=SHR,DSN=IDI.SIDIMLIB
//          DD DISP=SHR,DSN=IDI.SIDISLIB
//IPVTLIB   DD DISP=SHR,DSN=IDI.IDIPPROF
//          DD DISP=SHR,DSN=IPV.SIPVTENU19
//          DD DISP=SHR,DSN=IDI.SIDITLIB
//IPVPROF   DD DISP=SHR,DSN=IDI.IDIPPROF
```

---

### Installing the Fault Analyzer web interface

This feature is an optional feature.

The Fault Analyzer web interface is provided using a WebSphere Application Server Liberty profile. Delivered with Fault Analyzer is a PAX file, which when copied to HFS, contains all the necessary components to run the Liberty profile and the Fault Analyzer application.

The Fault Analyzer web interface installations steps:

1. “Running the sample job to copy server files into HFS”
2. “Applying the required RACF (or equivalent) rules”
3. “Customizing the server configuration” on page 447
4. “Customizing the sample Liberty procedures” on page 447
5. “Starting the Liberty profile” on page 448

### Running the sample job to copy server files into HFS

The sample job, IDISBRWS, which is located in the IDI.SIDISAM1 data set, copies the files required to run the Fault Analyzer web interface into an HFS directory of your choosing. Before running the job, you need to make any necessary changes, as described in the comment section of the job itself.

The job should now complete with a zero return code.

### Applying the required RACF (or equivalent) rules

The Fault Analyzer web interface requires a set of rules to be implemented in your z/OS security system. Many of these sample rules include placeholders where you can provide values tailored to your installation environment, including:

---

19. Data set created as part of the IBM Problem Determination Tools for z/OS Common Component (PDTCC) installation.

## Installing the Fault Analyzer web browser interface

### <ServerUserId>

The user ID that the two server jobs (started tasks) run under

### <UserId>

The user ID of someone who has been granted access to use the Fault Analyzer web interface (optional)

To allow the Fault Analyzer web interface to provide appropriate access to z/OS resources, the following RACF (or equivalent) rules must be applied:

```
RDEF SERVER BBG.ANGEL UACC(NONE)
PERMIT BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(<ServerUserId>)

RDEF SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(<ServerUserId>)

RDEF SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) ACCESS(READ) ID(<ServerUserId>)

RDEFINE SERVER BBG.SECPFX.IDIZDFLT UACC(NONE)
PERMIT BBG.SECPFX.IDIZDFLT CLASS(SERVER) ACCESS(READ) ID(<ServerUserId>)

SETROPTS RACLST(SERVER) GENERIC(SERVER) REFRESH
```

To facilitate user authentication and authorization, the following rules must be customized to suit your environment. The last two rules control user access to the Fault Analyzer web interface. Access can be granted to specific user IDs, or a UACC of READ can be used to give global access.

**Note:** Providing a UACC(READ) does not change the security checks made for individual history file data sets, it is just controlling access to the web application itself.

```
RDEFINE FACILITY BBG.SYNC.IDIZDFLT UACC(NONE)
PERMIT BBG.SYNC.IDIZDFLT ACCESS(CONTROL) CLASS(FACILITY) ID(<ServerUserId>)
SETROPTS RACLST(FACILITY) GENERIC(FACILITY) REFRESH

RDEFINE EJBROLE IDIZDFLT.IDIGWEB.AllAuthenticated UACC(NONE)
PERMIT IDIZDFLT.IDIGWEB.AllAuthenticated CLASS(EJBROLE) ACCESS(READ) ID(<UserId>)
SETROPTS RACLST(EJBROLE) GENERIC(EJBROLE) REFRESH

RDEFINE APPL IDIZDFLT UACC(NONE)
PERMIT IDIZDFLT CLASS(APPL) ACCESS(READ) ID(<UserId>)
SETROPTS RACLST(APPL) GENERIC(APPL) REFRESH
```

## Customizing the server configuration

By default, the Fault Analyzer web interface runs at the local IP address of 0.0.0.0 and port 9088. If these settings are unsuitable in the installation environment, then they can be modified in the ASCII settings file that is stored at:

- <ServerInstallDirectory>/wlp/usr/servers/FaultAnalyzer/bootstrap.properties

## Customizing the sample Liberty procedures

Two sample procedures, IDISBBGA and IDISBGGS, are provided in the IDI.SIDISAM1 data set. These sample procedures must be copied to an appropriate PROCLIB data set and customized as described in their comment sections.

## **Starting the Liberty profile**

The two Liberty profile started tasks must be started in a specific order:

1. IDISBBGA
2. IDISBBGS

These procedures can be started in a product, such as IBM SDSF, using the commands "/start IDISBBGA" and "/start IDISBBGS".

After starting both procedures, the message

[AUDIT ] CWWKF0011I: The server FaultAnalyzer is ready to run a smarter planet.

should be seen at the end of the IDISBBGS procedure output.

---

## **Part 3. Fault Analyzer reference information**



---

## Chapter 33. Options

You can use options to exert some control over the way that Fault Analyzer produces output. For example, there are options to:

- Change the fault analysis report contents.
- Change the action that Fault Analyzer takes at the time of the abend.

You can provide options at most stages of processing. If the option is not relevant to the current mode of processing (for example, you are trying to set the Exclude option when doing a batch reanalysis), it is disregarded. Fault Analyzer does not produce unnecessary warning messages in this situation.

Options can be set or changed in the following ways, listed in the order of their processing:

1. Product defaults provided by Fault Analyzer.
2. SMP/E USERMODS.  
For more information, see Chapter 16, “Customize Fault Analyzer by using USERMODS,” on page 253.
3. Configuration-options module IDIOPTLM.  
For more information, see Chapter 17, “Customize Fault Analyzer by using an IDIOPTLM configuration-options module,” on page 257.
4. Options located via an IDICNFUM user-options module.  
This module is only applicable to real-time analysis, and, if found, replaces step 5.  
For more information, see “User-options module IDICNFUM” on page 453.
5. Installation-wide defaults specified in the IDICNF00 parmlib member.  
The parmlib member is only read if a user-options module was not found in step 4.  
For more information, see “Parmlib member IDICNF00” on page 453.
6. Options that are specified in a user-options file through the \_IDI\_OPTSFILE environment variable.  
If found, replaces step 7.  
For more information, see “The \_IDI\_OPTSFILE environment variable” on page 454.
7. Options that are specified in a user-options file through the IDIOPTS DDname.  
Only read if a user-options file was not found in step 6.  
For more information, see “User options file IDIOPTS” on page 454.
8. Options that are specified in the JCL EXEC statement PARM field when performing batch reanalysis.  
For more information, see “The JCL EXEC statement PARM field” on page 455.
9. Options provided through the \_IDI\_OPTS environment variable.  
For more information, see “The \_IDI\_OPTS environment variable” on page 455.
10. Options set via the Analysis Control user exit.  
For more information, see “Analysis Control user exit” on page 368.

## Options

11. Settings of EPC data area fields with the End Processing user exit, as these might effectively override the RetainDump and MaxMinidumpPages options in effect.

For more information, see “End Processing user exit” on page 393.

If you do not specify an option, it takes either the product default (as indicated on the syntax diagram for each option), or has no value at all.

Some options can retain only one value. If more than one instance of such an option is specified, only the last occurrence has an effect. For example, if

```
PARM='Detail(LONG) Detail(SHORT)'
```

is specified, then the active option is Detail(SHORT).

Some options can have more than one value. These are, for example, the DataSets, Exits, Include, and Exclude options. The way in which they accumulate information is described for each option.

Wherever you specify an option, it is subject to these syntax rules:

- Only columns 1 - 71 are processed.
- Options can be specified anywhere in a line. They do not have to start in column 1.
- You can use a blank or a comma as a delimiter.
- Options can be continued across any number of lines, except when specified in the JCL EXEC statement PARM field, where the OS/390-imposed limit is 100 characters.
- Options specifications are not case-sensitive—all options are converted to uppercase.
- Comments are permitted anywhere and can be nested. The characters “/\*” identify the beginning of a comment, and “\*/” identify the end.

---

## Where to specify options

You can specify options in two files (IDICNF00 holds installation-wide default options, and IDIOPTS holds user options), or in the JCL EXEC statement PARM field. When options are set in the files, they are available to all modes of analysis.

For real-time analysis only, job-level substitution of installation-wide default options in IDICNF00 is available via a user-options module, IDICNFUM.

The JCL EXEC statement PARM field is only available for batch reanalysis.

Refer to “Batch reanalysis options” on page 95 and “Interactive reanalysis options” on page 103 for information about how to change options when performing fault reanalysis.

Regardless of where options are specified, an Analysis Control user exit is able to override the resulting settings. For details of this exit type, see “Analysis Control user exit” on page 368.

## Parmlib member IDICNF00

You can create a member IDICNF00 in SYS1.PARMLIB, or any other data set that is part of the logical parmlib concatenation. Optionally, a USERMOD can be applied to permit the use of another data set as explained in “Parmlib member IDICNFxx” on page 279.

The IDICNF00 member contains installation-wide default options that are read for every execution of Fault Analyzer.

For an example of a parmlib configuration member see Figure 137 on page 280.

## User-options module IDICNFUM

To replace the installation-wide default options during real-time analysis, you can create a user-options module containing the names of one or more partitioned data sets and members, each containing Fault Analyzer options. Fault Analyzer tries to open the data set members in the order of their specification. The first data set and member found to be available is used instead of the IDICNF00 parmlib member in the logical parmlib concatenation or in the alternative parmlib data set specified in the IDICNFDS CSECT.

The user-options module must be named IDICNFUM and must be a load module available via the standard MVS search path. If placed in a load library that is allocated to the JOBLIB DDname, this placement effectively provides job-level control of default options.

The load module can contain partitioned data set and member names only in standard MVS JCL syntax format:

*data-set-name (member-name)*

Each data set specification must be terminated by an X'00' byte. A second X'00' byte must follow the last data set specification to indicate the end of the list. If no data sets are specified, at least one X'00' byte is required.

For added flexibility in the use of user-options modules, the following symbolic names can be used in the specification of data set or member names:

### &SYSUID.

The user ID that is associated with the abending job or CICS transaction.

### &JOBNM.

The job name of the abending job.

### &PGMNM.

The program name on the EXEC statement in the abending job.

A job to create a sample user-options module is provided as member IDISCNFU in the softcopy samples data set.

The data set and member selected by Fault Analyzer is identified in message IDI0001I. If you want to see the data set and member names that were not selected (after substitution of variables), include the IDITRACE DDname in your job step. For example:

```
//IDITRACE DD SYSOUT=*
```

(See “IDITRACE under CICS” on page 307 for an alternative method of activating this trace under CICS.)

## Where to specify options

The name of the selected user-options module data set and member saved by Fault Analyzer in the history file and automatically reused as the default options file during reanalysis of the fault. For batch reanalysis, the data set and member is included in the generated JCL with the IDIBOPT DDname.

## The `_IDI_OPTSFILE` environment variable

Your application program can initialize an environment variable, `_IDI_OPTSFILE`, with an MVS data set (and optionally, member) name, or an HFS path and file name, containing options for Fault Analyzer, prior to abending or calling IDISNAP.

- **MVS data set name specification**

If specifying an MVS data set name, then the name must be immediately preceded by two forward slashes (//). The specified data set name is not case sensitive.

The specified user options file must be fixed 80-byte record length format and all options must be specified within columns 1 - 71.

- **HFS path and file name specification**

If specifying an HFS path and file name, then the name must not be preceded by two forward slashes.

The specified user options file is not subject to any particular attributes and options can be specified in any columns.

Example:

```
//* Sample BPXBATCH job
//RUN EXEC PGM=BPXBATCH, ...
:
//STDENV   DD * (select one of the following)
IDI_OPTSFILE=/my.SEQ.OPTIONS
IDI_OPTSFILE=/my.PDS.OPTIONS(faopts)
IDI_OPTSFILE=/u/fred/options_file
:
/*
/*
```

**Note:** No attempt is made to read options through the IDIOPTS DDname (see “User options file IDIOPTS”) if a user options file is found through the `_IDI_OPTSFILE` environment variable.

## User options file IDIOPTS

Fault Analyzer supports the specification of a user options file through the IDIOPTS DDname (CICS users, see “Specifying CICS options through the IDIOPTS DDname” on page 312).

Here is an example of a job that uses an in-stream user-options file to override any dump suppression if program MYAPPL abends:

```
//MYJOB1   JOB ...
//STEP1    EXEC PGM=MYAPPL
//SYSDUMP DD  DISP=SHR,DSN=MY.DUMP.DATA.SET
//IDIOPTS  DD  *
      RetainDump(ALL) /* do not suppress the dump if MYAPPL
                        abends */
/*
/*
```

*Figure 175. Sample job specifying user-options file*

The user options file must be fixed 80-byte record length format and all options must be specified within columns 1 - 71.

**Note:** If a user options file is found through the \_IDI\_OPTSFILE environment variable (see “The \_IDI\_OPTSFILE environment variable” on page 454), then no attempt is made to read options through the IDIOPTS DDname.

## The JCL EXEC statement PARM field

You can specify options in the JCL EXEC statement PARM field when executing Fault Analyzer in batch reanalysis mode. For example:

```
//MYJOB2    JOB
//STEP1      EXEC PGM=IDIDA,
//  PARM='/Detail(LONG),PreferredFormattingWidth(132)'
  :
```

## The \_IDI\_OPTS environment variable

Your application program can initialize an environment variable, \_IDI\_OPTS, with options for Fault Analyzer prior to abending or calling IDISNAP.

Example:

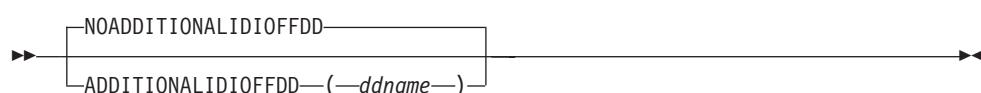
```
/* Sample BPXBATCH job
//RUN EXEC PGM=BPXBATCH, ...
  :
//STDENV   DD *
  _IDI_OPTS=DataSets(IDIHIST(MY.HIST)),Detail(L)
  :
  /*
  :
```

## Option descriptions

The following explains each option in detail.

### AdditionalIDIOffDD

#### Syntax



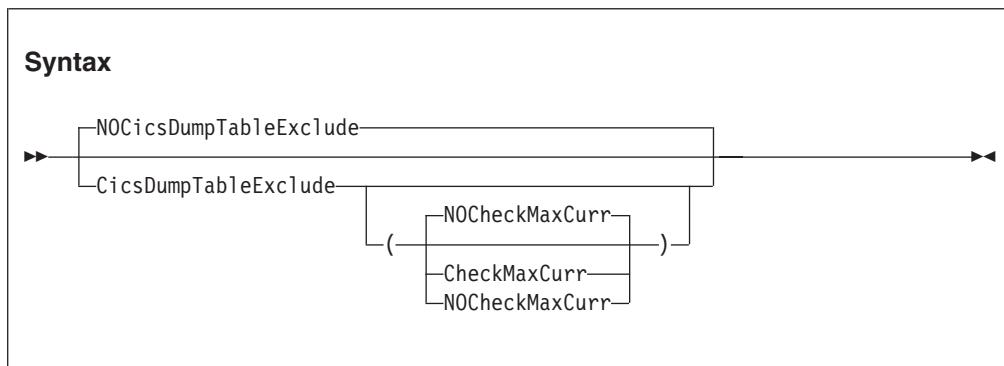
The AdditionalIDIOffDD option can be used to provide an extra DDname, which is equivalent to the normal Fault Analyzer IDIOFF DDname (for details, see “Turning off Fault Analyzer with a JCL switch (IDIOFF)” on page 356). If specified, any job with *ddname* allocated does not invoke Fault Analyzer for real-time abend processing.

This option is read by the Fault Analyzer IDIS subsystem, and the setting made available to abending regions via cross memory access. Changes to the AdditionalIDIOffDD option only take effect after stopping and restarting the Fault Analyzer IDIS subsystem (for details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239).

This option is not included in the section of the fault analysis report that shows options in effect.

## CICSDumpTableExclude

### CICSDumpTableExclude



The CICSDumpTableExclude option excludes a CICS transaction abend from Fault Analyzer real-time processing if the CICS transaction dump table action for the same abend code specifies NOTRANDUMP.

If excluded, then message IDI0101I is issued, no analysis report is produced, and no fault entry is written to the history file.

Exclusion of Fault Analyzer processing based on this option precedes any other Fault Analyzer methods of preventing analysis, such as the Exclude or NoDup options (see “Real-time exclusion processing” on page 27).

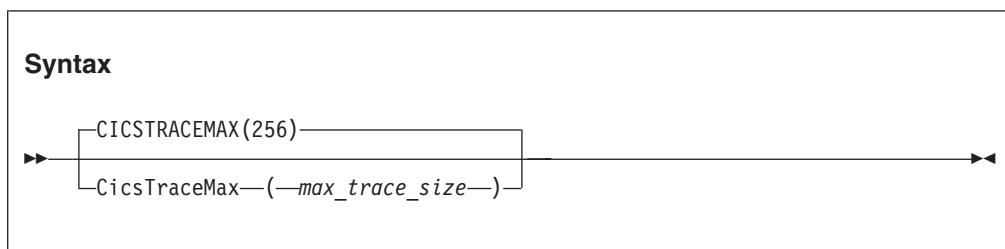
If the CheckMaxCurr suboption is specified, then Fault Analyzer compares the current number of CICS dump requests for a specific dump code to the maximum setting for that dumpcode. If the current value exceeds the maximum value, then Fault Analyzer analysis is skipped and message IDI0180I is issued.

The default setting is NOCheckMaxCurr, in which case analysis continues regardless of the maximum setting for the dumpcode.

When NOCICSDumpTableExclude is in effect (default), then Fault Analyzer does not use the CICS transaction dump table in its exclude checking.

This option is not included in the section of the fault analysis report that shows options in effect.

## CICSTraceMax



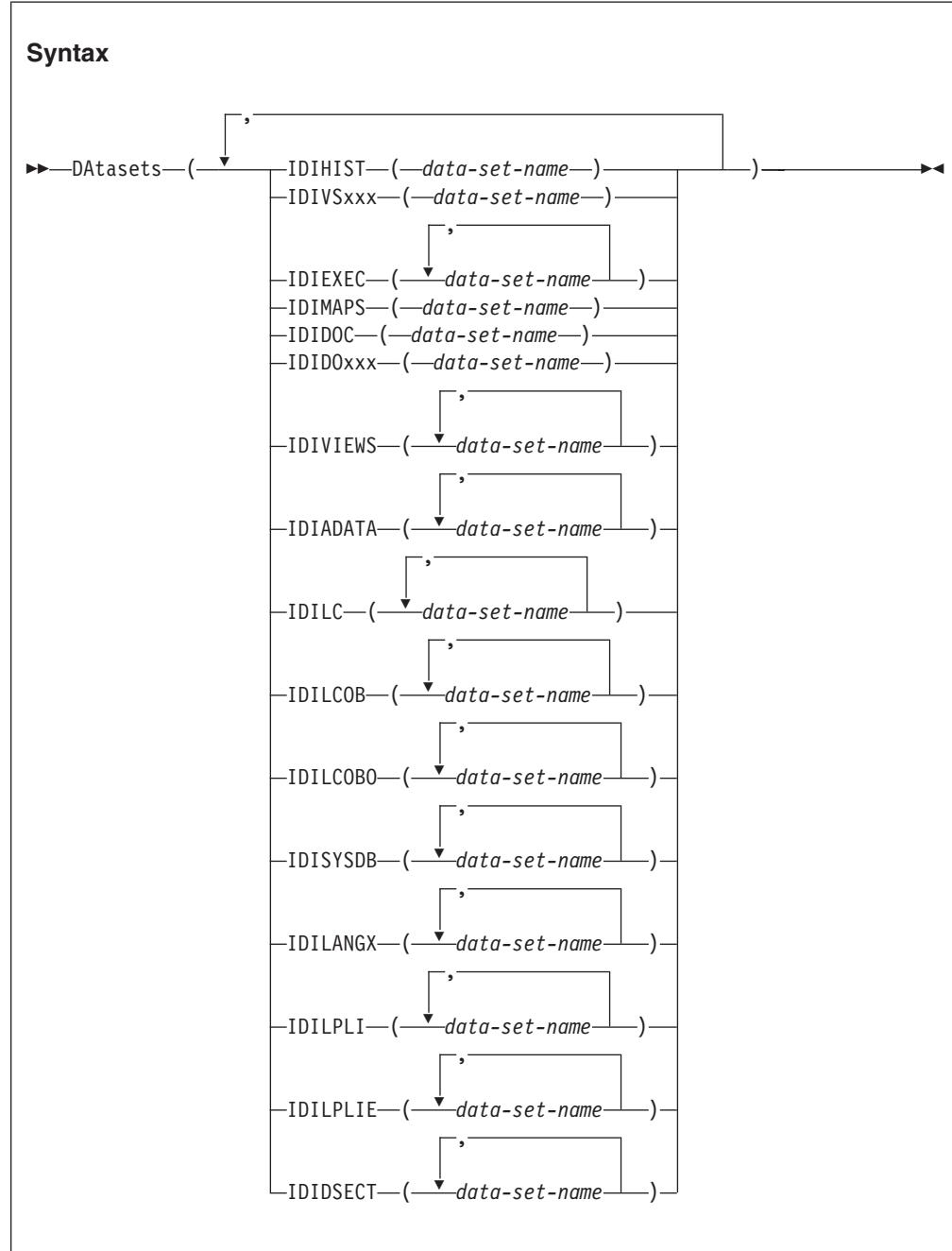
The CICSTraceMax option is applicable to CICS system dump analysis only, and is used to specify the maximum CICS trace table size in kilobytes which can be included in a fault entry minidump.

The value of *max\_trace\_size* must be between 0 and 2097151 kilobytes, both inclusive.

If not specified, the CICSTraceMax option value defaults to 256 kilobytes.

If the actual CICS trace size exceeds the CICSTraceMax option value in effect, then the trace table is read from the associated CICS system dump during fault entry reanalysis.

## DataSets



## DataSets

The DataSets option specifies as suboptions the DDnames and their associated data set names that are to be dynamically allocated by Fault Analyzer.

### **IDIHIST**

The name of the PDS(E) history file where the fault entry is to be or was written. Default value is IDI.HIST.

### **IDIEXEC**

The name of one or more PDS(E) data sets containing REXX user exits.

### **IDIMAPS**

The name of the PDS(E) data set containing data area mappings provided with Fault Analyzer. Default value is IDI.SIDIMAPS.

### **IDIDOC**

The name of the distributed book index and override PDS(E) data set. Default value is IDI.SIDIDOC1.

### **IDIDOxxx**

The name of an extra distributed book index and override PDS(E) data set for a multicultural support feature, where xxx is a valid language ID for the Language option, other than ENU—for example, IDIDOJPN. The data set specified is only used when the equivalent Language option is in effect. No default value is provided.

### **IDIVSxxx**

The name of the VSAM KSDS message and abend code explanation repository, where xxx is a valid language ID for the Language option—for example, IDIVSENU. For values of xxx other than ENU, the data set specified is only used when the equivalent Language option is in effect. Default value is IDI.IDIVSENU.

### **IDIVIEWS**

The name of one or more PDS(E) data sets containing members defining the fault history files to be viewed in a single ISPF display.

### **IDIADATA**

The name of one or more sequential or PDS(E) data sets holding Assembler SYSADATA files.

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

### **IDILC** The name of one or more sequential or PDS(E) data sets holding C compiler listings.

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

### **IDILCOB**

The name of one or more sequential or PDS(E) data sets holding COBOL compiler listings (other than OS/VS COBOL).

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

### **IDILCOBO**

The name of one or more sequential or PDS(E) data sets holding OS/VS COBOL compiler listings.

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

**IDISYSDB**

The name of one or more sequential or PDS(E) data sets containing COBOL or Enterprise PL/I SYSDEBUG side files, or XL C/C++ MDBG side files. (These side files are created when compiling a COBOL program with the TEST(,,SEPERATE) option. MDBG side files are created using the CDADBLGLD utility.)

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

**IDILANGX**

I

The name of one or more sequential or PDS(E) data sets holding LANGX side files.

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

**IDILPLI**

The name of one or more sequential or PDS(E) data sets holding PL/I compiler listings (other than Enterprise PL/I).

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

**IDILPLIE**

The name of one or more sequential or PDS(E) data sets holding Enterprise PL/I compiler listings.

For details about required data set attributes for this DDname, see “Compiler listings and side file attributes” on page 296.

**IDIDSECT**

The name of one or more PDS(E) data sets, containing assembler macro or DSECT copybooks that are to be used with the interactive reanalysis DSECT command. For details, see “Mapping storage areas using DSECT information” on page 154.

IDIHIST, IDIEXEC, IDIMAPS, IDIDOC, IDIDOxxx, and IDIVSxxx have only one data set name value, and a second specification replaces, rather than accumulates. Multiple specifications of the other DataSets suboptions are cumulative and all the data sets, wherever they have been specified, are included in the final logical concatenation of the respective DDname.

To specify a DUMMY data set for any DDname, a data set name of NULLFILE can be used.

The names of compiler listing or side file data sets used during real-time analysis are saved with the fault entry in the history file and are automatically used if reanalysis is performed. Therefore, there is generally no need to specify data sets using the DataSets option for reanalysis, unless a compiler listing or side file that was not available during real-time analysis is to be made available.

**Data set logical replacement or concatenation order**

The logical replacement or concatenation order of Fault Analyzer data sets is shown in the following:

1. Any data sets provided by an Analysis Control user exit (see “Analysis Control user exit” on page 368 for details).
2. Any explicitly coded JCL statements for the DDname.

## Data set logical replacement or concatenation order

3. Data sets from all DataSets options that are specified in the JCL EXEC statement PARM field (reanalysis only).
4. Data sets from all DataSets options that are specified in the user options file.
5. Data sets from all DataSets options that are specified in the parmlib configuration member.

These options include the use of a system-wide alternate parmlib data set (for details, refer to "Parmlib member IDICNFxx" on page 279.) and a data set specified via a user-options module (for details, refer to "User-options module IDICNFUM" on page 453.).

### Dropping IDICNF00 parmlib member data set specifications

It is possible to drop all data set specifications for a given DDname specified in the IDICNF00 parmlib member by using the special data set name, -DROPCNF-.

If, for example, the IDICNF00 parmlib member contained:

```
DataSets(IDILCOB(FRED.L1,FRED.L2))
```

and an IDIOPTS user-options file contained:

```
DataSets(IDILCOB(FRED.L3,-DROPCNF-,FRED.L4))
```

then the final logical concatenation of data sets for the IDILCOB DDname would be:

```
IDILCOB DD DISP=SHR,DSN=FRED.L3  
          DD DISP=SHR,DSN=FRED.L4
```

Note that the -DROPCNF- data set name can be specified anywhere that a DataSets option can be specified. It can, however, not be provided by an Analysis Control user exit.

### Data set name substitution symbols

All data set names that are specified with the DataSets option can contain standard MVS symbols<sup>20</sup>. These are resolved prior to any DDname-specific symbol substitution being performed by Fault Analyzer, as detailed below.

In the following, an X indicates permitted symbols that can be used as part of the data set name(s) for each DDname:

Table 11. Permitted data set name substitution symbols by DDname

DDname	&PGM.	&SYSUID.	&TSOPFX.	&USERID.	&SYSNAME.
IDIHIST					X
IDIVSxxx					X
IDIEXEC				X (Note 1)	X
IDIMAPS					X
IDIDOC					X
IDIDOxxx					X
IDIVIEWS (Note 2)		X	X		X
IDIADATA	X	X (Note 3)	X (Note 3)	X	X
IDILC	X	X (Note 3)	X (Note 3)	X	X
IDILCOB	X	X (Note 3)	X (Note 3)	X	X

20. The available symbols on a given MVS system can be displayed with the MVS operator command "D SYMBOLS".

## Dropping IDICNF00 parmlib member data set specifications

*Table 11. Permitted data set name substitution symbols by DDname (continued)*

DDname	&PGM.	&SYSUID.	&TSOPFX.	&USERID.	&SYSNAME.
IDILCOBO	X	X (Note 3)	X (Note 3)	X	X
IDISYSDB	X	X (Note 3)	X (Note 3)	X	X
IDILANGX	X	X (Note 3)	X (Note 3)	X	X
IDILPLI	X	X (Note 3)	X (Note 3)	X	X
IDILPLIE	X	X (Note 3)	X (Note 3)	X	X
IDIDSECT				X	X

Notes:

- 1 No value is available for the &USERID. variable when performing MVS dump analysis (see “File->Analyze MVS Dump Data Set” on page 61).
- 2 These symbols can be used in the data set names that are specified in the IDIVIEWS suboption of the DataSets option, as well as in the data set names that are specified in the individual view members contained within the IDIVIEW data sets—for details, see “Setting up views” on page 263.
- 3 Substitution of these symbols is only performed during interactive reanalysis.

If, at the time of performing symbol value substitution, a value for a symbol is unavailable, then any data set names that include that symbol are ignored. An example of this ignoring is the &USERID. variable mentioned in the preceding note.

All symbol names specified must include the ending period, unless the symbol name is at the very end of a data set name. The symbol name, including the leading ampersand (&) and the ending period is replaced by the symbol value. Hence, if a symbol name is specified immediately ahead of a data set qualifier delimiter (period), then two consecutive periods must be specified, for example "&SYSUID..LISTINGS".

Data set names containing symbols do not cause errors during Fault Analyzer processing if, after their substitution, the data sets cannot be found, unless the data set is critical to the analysis.

Substitution symbols:

### &PGM.

All instances of this symbol are substituted by the current program name when Fault Analyzer is searching for compiler listings or side files. To use this functionality, compiler listings or side files in sequential data sets can be stored with a naming convention that includes the program name.

For example, if the compiler listings for programs P1 and P2 are stored in the data sets FRED.LISTING.P1 and FRED.LISTING.P2 respectively, then these can both be located by Fault Analyzer by specifying "FRED.LISTING.&PGM." for the appropriate DDname in the DataSets option.

### &SYSUID.

All instances of this symbol are substituted by the TSO user ID under which the Fault Analyzer ISPF interface is being used. By using this symbol in data set names that are specified in the IDICNF00 parmlib

## Dropping IDICNF00 parmlib member data set specifications

member, automatic user-specific access to private data sets can be provided, given that an appropriate data set naming convention is adhered to by the installation.

For example, specify the option

```
DataSets(IDIVIEWS(&SYSUID..VIEWS,PROD.VIEWS))
```

in the IDICNF00 parmlib member. Then all users of the Fault Analyzer ISPF interface for whom a data set named *user-id*.VIEWS exist, where *user-id* is the user's TSO user ID, can place private Fault Analyzer View definitions in this data set. These definitions either complement any installation-wide definitions in the PROD.VIEWS data set, or override them.

### &TSOPFX.

All instances of this symbol are substituted by the TSO profile prefix for the user under which the Fault Analyzer ISPF interface is being used. By using this symbol in data set names that are specified in the IDICNF00 parmlib member, automatic user-specific access to private data sets can be provided, given that an appropriate data set naming convention is adhered to by the installation.

For example, specify the option

```
DataSets(IDIVIEWS(&TSOPFX..VIEWS,PROD.VIEWS))
```

in the IDICNF00 parmlib member. Then all users of the Fault Analyzer ISPF interface for whom a data set named *tso-prefix*.VIEWS exist, where *tso-prefix* is the user's TSO profile prefix, can place private Fault Analyzer View definitions in this data set. These definitions either complement any installation-wide definitions in the PROD.VIEWS data set, or override them.

### &USERID.

All instances of this symbol are substituted by the user ID under which the abend occurred.

### &SYSNAME.

All instances of this symbol are substituted by the system name from CVTSNAME, as at the time of substitution.

**Symbol substring specification:** Substring specification of all symbols (MVS system symbols, as well as Fault Analyzer specific symbols) is permitted.

#### Syntax

```
►-& —symbol_name—(—start—  
                  [ :number ])—.————►
```

where:

#### symbol\_name

The name of the symbol, for example USERID.

**start** The character position where the substring is to start. The first character in the original symbol is position 1, the second is position 2, and so on. If *start* is positive, the system counts from the starting position to the end of

## Dropping IDICNF00 parmlib member data set specifications

the string. If *start* is negative (in other words, a minus sign appears before it), the system counts backwards from the ending position of the string.

### number

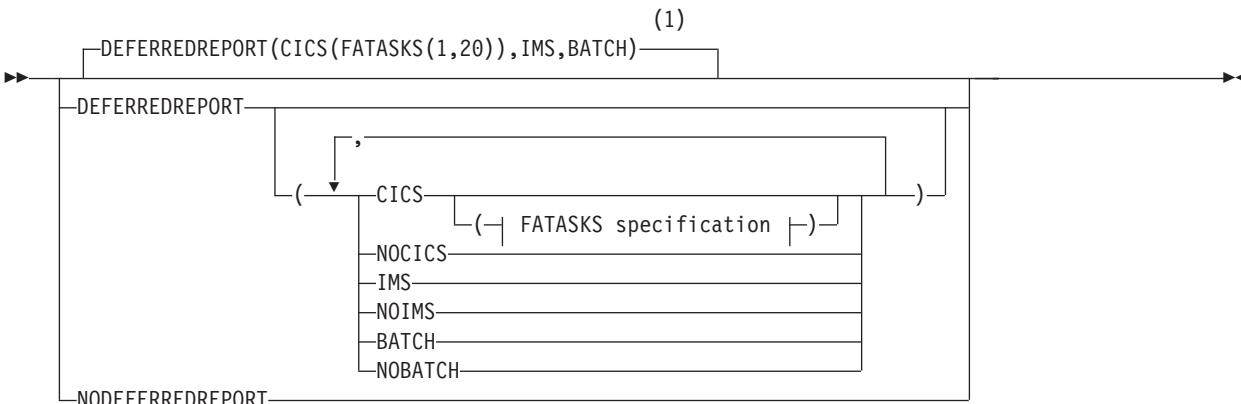
The number of characters, from the starting position to the ending position of the string, that the substring is to contain. If *number* is not specified, the substring length defaults to 1. Do not specify a negative number for *number*.

For example, if the substitution value for symbol &PGM. contains MYPROG1, and you want to limit the qualifier to the first six characters, you can specify &PGM(1:6).

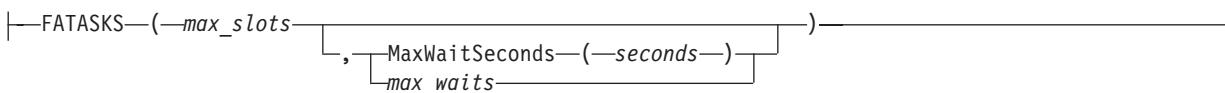
Refer to *MVS Initialization and Tuning Reference* section "Using substrings of system symbols" for complete information about symbol substring specification. Note that the use of double ampersand notation is not supported.

## DeferredReport

### Syntax



### FATASKS specification:



### Notes:

- 1 The product default, when no DeferredReport option has been specified, is DEFERREDREPORT(CICS(FATASKS(1,20)),NOIMS,NOBATCH).

This option applies to real-time analysis only, and can be used when real-time performance is critical. When used, no real-time analysis report is produced, but a fault entry is written that can later be reanalyzed.

For compatibility with earlier versions of Fault Analyzer, specification of  
DeferredReport(CICSFATasks(max\_slots<,max\_waits>))

## DeferredReport

is permitted until further notice. This specification is equivalent to  
`DeferredReport(CICS(FATasks(max_slots<,max_waits>)))`

The execution environment under which the DeferredReport option is enabled is specified using the CICS | NoCICS, IMS | NoIMS, or Batch | NoBatch suboptions.

The product default, when the DeferredReport option is not specified, is CICS only. The default, when the DeferredReport option is specified without any suboptions, is all execution environments.

As far as the DeferredReport option is concerned, a CICS transaction that also interfaces with IMS, is controlled using the CICS suboption. Hence, specification of `DeferredReport(NoCICS,IMS)`

disables the DeferredReport option for such an application, whereas  
`DeferredReport(CICS,NoIMS)`

enables it.

When used with CICS, the optional FATASKS suboption can be specified:

*max\_slots*

Specifies the maximum number of execution slots that are made available. Each execution slot permits one instance of Fault Analyzer real-time analysis to run, effectively enabling parallel execution or multi-tasking.

The valid range is 1 - 6.

The default is 1.

If NODeferredReport is specified, then the maximum number of execution slots is set to 1.

**MaxWaitSeconds(*seconds*)**

Specifies the maximum number of seconds that a fault is allowed to be queued waiting for analysis. If a fault has been waiting for longer than the current limit in effect, then message IDI0132W is issued and the analysis of that fault is skipped, thus requiring normal CICS transaction dump analysis to be performed (that is, not using Fault Analyzer).

The waiting queue length is always 20 when this suboption is specified. The behavior described for the *max\_waits* suboption is still applicable to faults that occur when 20 queued up faults are already waiting.

The valid range is 0 - 3600. If 0 is specified, then no time limitation is imposed.

The default is 0.

Using MaxWaitSeconds(*seconds*) instead of *max\_waits* is recommended.

*max\_waits*

Specifies the maximum number of faults allowed to be queued for analysis. If the maximum has already been reached when another fault occurs, then message IDI0118W is issued and the analysis of that fault is skipped, thus requiring normal CICS transaction dump analysis to be performed (that is, not using Fault Analyzer).

The valid range is 1 - 20.

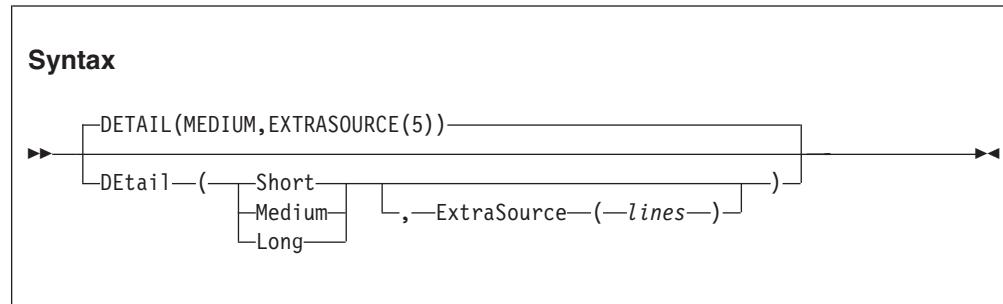
The default is 20.

**Note:** It might be desirable to reduce *max\_waits* to a lesser value if analysis is slow in a CPU constrained environment. This reduction could have the effect of abends being rejected from analysis with the IDI0118W message, but that might be preferred during high abend activity to having the task wait for abend processing in an over-committed environment.

#### Note:

1. If DeferredReport is in effect, and the MaxMinidumpPages limit is exceeded, then the DeferredReport option is overridden and a report is written. If this situation occurs, then message IDI0133W is issued and a note is added against the DeferredReport option in the "Options in Effect" section of the analysis report. To prevent frequent occurrences of this situation, ensure that the MaxMinidumpPages limit is adequate.
2. Although fault entries written with the DeferredReport option in effect do not initially contain a saved report, one is added the first time the 'V' line command is used from the Fault Entry List display, given that the user performing this action has update access to the history file.
3. This option (but not the FATASKS suboption) can be modified or set by an Analysis Control user exit—for details, see “Analysis Control user exit” on page 368.
4. For maximum Fault Analyzer performance, \$\$INDEX member caching in the IDIS subsystem should also be considered. For details, see “Caching of history file \$\$INDEX data” on page 240.

## Detail



The Detail option specifies the level of detail that should be included in the fault analysis report as either SHORT, MEDIUM, or LONG.

If more (or less) than the default of 5 extra source lines, before and after the source line for an event, should be shown in the Fault Analyzer report, then the ExtraSource(*lines*) suboption can be used. The ExtraSource(*lines*) suboption affects the number of source lines shown in the detailed event information section only.

The complete Fault Analyzer fault analysis report contains the following sections:

- Fault Analyzer synopsis
- Fault Analyzer summary
- Detailed information about individual events
- Information not directly related to any event, such as console messages
- Information about the abending job
- Fault Analyzer options in effect

## Detail

Depending on the specification of the Detail option, all or parts of the complete report are produced:

### Detail(SHORT)

- All sections of the report are included, except for the system-wide information. Detailed information is only included for the point of failure event.
- Summarized CICS trace is included (when applicable),

### Detail(MEDIUM)

- All sections of the report are included. However, detailed information is only included for events up until (and including) the first abend event that is also the point-of-failure event, or which follows the point-of-failure event. This option is the default.
- Summarized CICS trace is included (when applicable).

### Detail(LONG)

- All sections of the report are included and detailed information is provided for all events.
- Full CICS trace is included (when applicable).
- 4K of storage around each user-code event general purpose register is included in the minidump.

This option does not apply to interactive reanalysis.

## DumpDSN

### Syntax

```
►--DUMPDSN--(—dump-data-set-name—)-----►  
| DSN |
```

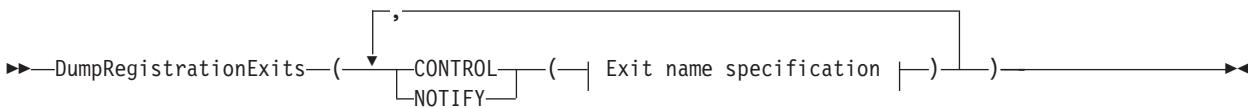
Use the DumpDSN option when an abend caused a SYSMDUMP to be written to a dump data set and you want to reanalyze the dump using your own JCL. The option specifies the dump data set against which fault analysis is to be performed.

This option applies only to batch reanalysis and is ignored if specified in an IDICNFxx parmlib member.

If you initiate the reanalysis from the fault history file, then Fault Analyzer provides the dump data set name automatically.

The logical record length of the SYSMDUMP data set must be 4160 if it does not contain ASA printer control characters, or 4161 if it does. Typically, only SYSMDUMP data sets that have been extracted from the JES SPOOL via SDSF contain ASA printer control characters.

## DumpRegistrationExits

**Syntax****Exit name specification:**

The DumpRegistrationExits option specifies the types and names of user exits to be invoked during MVS SVC dump registration processing (IDIXTSEL).

The dump registration Analysis Control and Notification user exits are driven as the SVC dump data set name is recorded in a history file fault entry for later analysis by the end user. The SVC dumps might have been triggered by abends or Fault Analyzer recovery fault recording processing. The IDIXTSEL process requires the IDI.SIDISAM1(IDIWTSEL) sample ++USERMOD to be installed (for details, see “Installing the MVS post-dump exit IDIXTSEL” on page 313).

The MVS SVC dump registration Analysis Control and Notification user exits run from the Fault Analyzer IDIS subsystem. Therefore, the DumpRegistrationExits option must be specified in the IDICNFxx parmlib member, or via an IDIOPTS DD statement in the IDIS subsystem JCL. The DumpRegistrationExits option is ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS region or batch job.

Changes to the DumpRegistrationExits option only take effect after stopping and restarting the Fault Analyzer IDIS subsystem (for details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 239).

Multiple specifications of the DumpRegistrationExits option are cumulative.

Exits can be either REXX EXECs or load modules:

- REXX EXECs must be specified as  
REXX(*exit\_name\_1*, *exit\_name\_2*, ...)

and be available via the IDIEXEC DDname.

- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

**CONTROL**

Analysis Control user exit. This exit can be used to modify options in effect. For details, see “Analysis Control user exit (MVS SVC Dump registration)” on page 371.

## DumpRegistrationExits

### NOTIFY

Notification user exit. This exit can, for example, be used to provide installation-specific notification of recorded faults. For details, see "Notification user exit (MVS SVC Dump registration)" on page 402.

The exit name that is specified as *exit\_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

### NONE

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

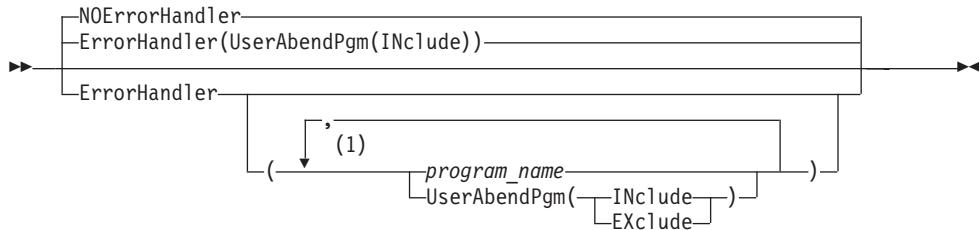
### -DROPCNF-

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNF00 parmlib member. For details, see "Dropping IDICNF00 parmlib member user exit specifications" on page 475.

Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

## ErrorHandler

### Syntax



### Notes:

- 1 Either comma or blank character is permitted as delimiter.

The ErrorHandler option can be used to specify one or more common error handler program or CSECT names, which should not be selected as the point-of-failure event in the Fault Analyzer report. Instead, the next earlier user-code event becomes the point-of-failure event, and thus be used in duplicate fault determination, and so on.

The following suboptions can be specified:

#### *program\_name*

One or more program or CSECT names of common error handlers, which are not selected as the point-of-failure event.

#### UserAbendPgm(INclude | EXclude)

By default, an event for a user abend is not selected as the point-of-failure. This is the equivalent to specifying UserAbendPgm(Include). To allow user

abend events to be selected as the point-of-failure event, subject to the name not appearing in the program\_name list, specify UserAbendPgm(Exclude).

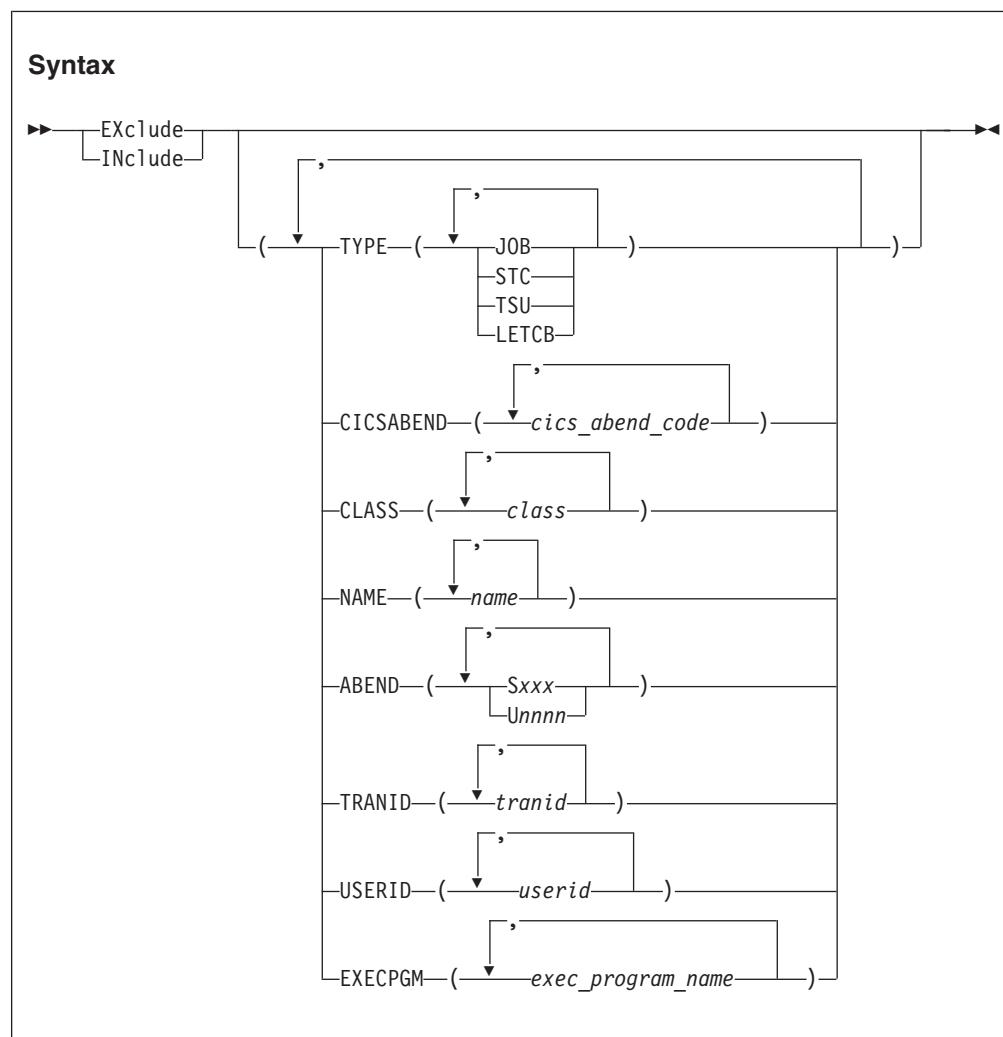
**Note:** If no earlier user-code event can be determined, then it is possible that the error handler program event is still designated as the point of failure.

The last specification of the ErrorHandler or NOErrorHandler option is in effect. Subsequent specifications override any previous specification completely, that is, the program or CSECT names are not cumulative.

## Exclude/Include

The Exclude and Include options are complimentary processes, sharing common parameters to control which job exceptions should be processed by Fault Analyzer. The Exclude/Include process, as described in “Controlling which jobs are analyzed with Exclude processing” on page 281, should be read and understood before studying this section on the parameters.

The term "work unit" is used in this section to refer to either a batch job, a started task, or a TSO user.



## Exclude/Include

where:

**TYPE** Specifies the type of work unit as either JOB (batch jobs), STC (started tasks), or TSU (TSO users).

An extra type, LETCB, specifies that LE must be active for the abend TCB. This type can be used to distinguish between abends that occur in application tasks, as opposed to system tasks, provided that the application is written in a language that uses Language Environment.

### CICSABEND

Specifies one or more abend codes for CICS transactions as *cics\_abend\_code*. Each abend code must be four alphanumerical characters.

The abend code tested is the final abend code for the transaction.

Note that CICS system dumps captured via the IEAVTSEL post-dump exit, IDIXTSEL, are not affected by the Exclude/Include option.

### CLASS

Specifies one or more execution classes for batch jobs as *class*.

### NAME

Specifies the names of one or more jobs, tasks, or TSO users as *name*.

### ABEND

Specifies one or more system or user abend codes as one of the following:

- **Sxxx**  
where *xxx* is a three-character hexadecimal system abend code (for example, S0C4)
- **Unnnn**  
where *nnnn* is a four-character decimal digit user abend code (for example, U4039)

The abend code tested is the final abend code for the abending job step.

### TRANID

Specifies the names of one or more CICS transactions as *tranid*.

### USERID

Specifies the TSO or CICS user ID, or the user ID under which a batch job, a CICS transaction, or a started task is executing, as *userid*.

### EXECPGM

Specifies the program name from the JCL EXEC statement PGM keyword, as *exec\_program\_name*.

**Note:** The SYSABEND suboption, which has been replaced by the ABEND suboption, is supported for backwards compatibility only.

When an abending task meets the Exclude criteria, and no subsequent Include criteria also matches the task, the abend is not logged in the history file and no further Fault Analyzer processing is performed.

Specification rules:

- Individual suboptions, and values within suboptions, must be delimited by either one or more blank characters, or a comma.
- Wildcards are permitted in the specification of criterion values. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character.

For examples of using wildcards with criterion values, see “Exclude/Include wildcard examples” on page 472.

- If no Exclude criteria are specified, the default is to include everything.
- All suboptions of an Exclude or Include criterion (that is, TYPE, CICSABEND, CLASS, NAME, ABEND, TRANID, USERID, and EXECPGM) must be satisfied for the criterion to be met (logical AND). It is possible to create Exclude or Include criteria that are never met, for example

```
Exclude(TYPE(STC) CLASS(A))
```

The reason why these criteria are never met is that a started task does not run in a JES initiator address space, and therefore is not associated with a particular class.

In this case, and if no other Exclude criteria are specified (anywhere), then no job would be excluded, which means that Fault Analyzer analyzes any abends that occur.

- If more than one *type* (JOB, STC, TSU, or LETCB), *cics-abend-code*, *class*, *name*, *abend-code* (Sxxx or Unnnm), *tranid*, *userid*, or *exec-program-name* value is specified within a single TYPE, CICSABEND, CLASS, NAME, ABEND, TRANID, USERID, or EXECPGM suboption, then a match on any one value is sufficient for the entire suboption to match (logical OR).
- If multiple Exclude options are specified, then exclusion occurs if the criteria matches for any one, provided that a matching Include criteria does not follow.

This option does not apply to batch or interactive reanalysis.

This option is not included in the section of the fault analysis report that shows options in effect.

**Note:** Every Include and Exclude criteria is checked against the abending task without regard to any previous Include or Exclude criteria. Hence, it is important to ensure that the order of these criteria in the parmlib config member, and, if available, in the user options file, result in the desired installation-specific rule set. For example, to exclude all batch jobs, except those executing in class A, you could specify the following sequence of criteria:

```
Exclude          /* This excludes everything */
Include(CLASS(A)) /* This includes batch jobs in class A only */
```

For more information, see “Controlling which jobs are analyzed with Exclude processing” on page 281.

Another way to stop Fault Analyzer from analyzing a fault is to use the IDIOFF DD statement switch. For details, see “Turning off Fault Analyzer with a JCL switch (IDIOFF)” on page 356.

## Using Exclude/Include options with dump registration processing

While Exclude/Include options are also applicable to dump registration processing, they are not very useful since not much is known about the conditions which resulted in the dump. This lack of knowledge is due to the dump being written by the dump services address space (for details, see “Dump registration processing” on page 27), and Fault Analyzer therefore invoked here, instead of in the address space which issued the dump.

It is therefore recommended that the dump registration Analysis Control user exit (see “Analysis Control user exit (MVS SVC Dump registration)” on page 371) is

## Exclude/Include

instead used to control the inclusion or exclusion of dump registration processing, primarily based on information in the ENV data area (for details, see “ENV - Common exit environment information” on page 513).

In the ENV data area, the field JOB\_TYPE is set to 'D' for dump registration processing.

### Exclude/Include trace information

By adding an IDITRACE DDname to your job, Fault Analyzer can provide you with trace information that might help you understand why a particular fault is being included or excluded from real-time analysis. For example:

```
//IDITRACE DD SYSOUT=*
```

(See “IDITRACE under CICS” on page 307 for an alternative method of activating this trace under CICS.)

The trace information is written to the IDITRACE DDname destination. An example of an Exclude/Include option-processing trace follows:

---

```
EXCLUDE/INCLUDE option criterion match values from abending job:  
ABEND. . . . . : SOCB  
CICSABEND. . . . . : n/a  
CLASS. . . . . : A  
EXECPGM. . . . . : TEST1  
NAME . . . . . : ICC20010  
TRANID . . . . . : n/a  
TYPE . . . . . : JOB,LETCB  
USERID . . . . . : FRED  
Parmlib config member EXCLUDE(TYPE(TSU)) option did not match; Exclude/include status is: INCLUDE  
Parmlib config member INCLUDE(USERID(FRED)) option matched; Exclude/include status is: INCLUDE  
Parmlib config member EXCLUDE(TYPE(STC) NAME(VTAM DFHSM)) option did not match; Exclude/include status is: INCLUDE  
Parmlib config member EXCLUDE(SYSABEND(*37)) option did not match; Exclude/include status is: INCLUDE
```

---

Figure 176. Sample Exclude/Include option-processing trace

Trace information is not available for CICS transactions.

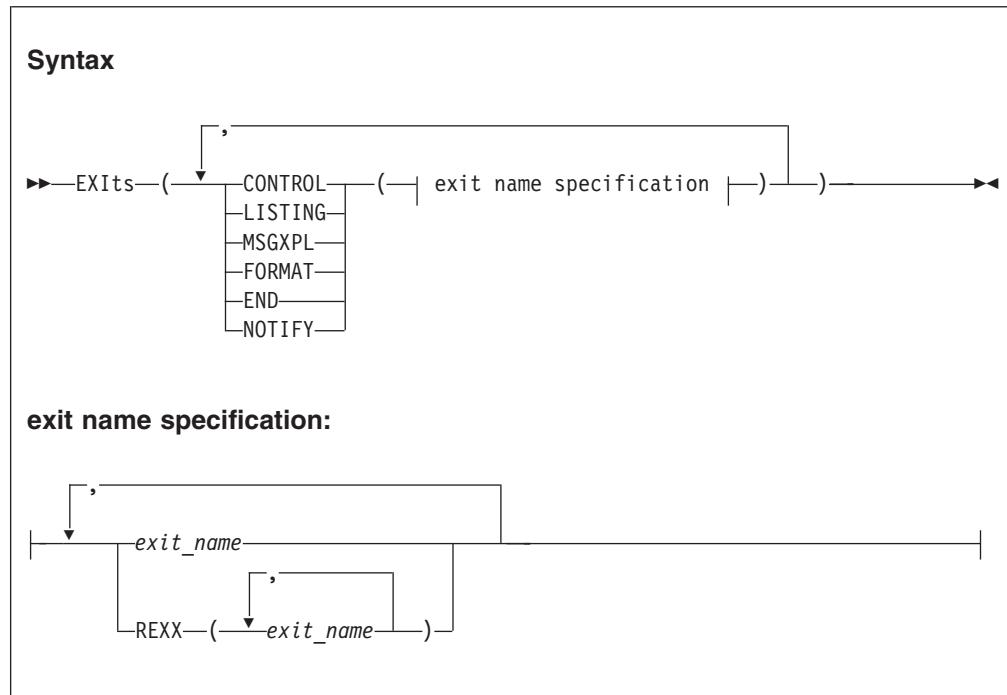
### Exclude/Include wildcard examples

The following are examples of Exclude/Include criterion values with wildcards:

Table 12. Wildcard examples

Criterion Value	Compare Value	Match
ABCD*	ABC	No
ABCD*	ABCD	Yes
ABCD*	ABCDE	Yes
A*CD	ABC	No
A*CD	ABCD	Yes
A*CD	ABCDE	No
A*DE	ABCDE	Yes
A**DE	ABCDE	Yes
A%DE	ABCDE	No
AB%DE	ABCDE	Yes
%B*	ABCDE	Yes

## Exits



The Exits option specifies the types and names of user exits to be invoked during Fault Analyzer execution. Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

**Note:** For information about specifying user exits for the IDIUTIL batch utility, see “EXITS control statement” on page 343 instead.

Multiple specifications of the Exits option are cumulative.

Exits can be either REXX EXECs or load modules (note that REXX is the only supported programming language for the Formatting user exit):

- REXX EXECs must be specified as  
REXX(*exit\_name\_1*, *exit\_name\_2*, ...)

and be available via the IDIEXEC DDname.

- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

### CONTROL

Analysis Control user exit. This exit can be used to modify options in effect or exclude a fault from analysis. For details, see “Analysis Control user exit” on page 368.

### LISTING

Compiler Listing Read user exit. This exit can be used to obtain source code information from sources other than compiler listings or Fault Analyzer side files that are contained in available MVS PDS (or PDSE) data sets. For details, see “Compiler Listing Read user exit” on page 372.

## Exits

### MSGXPL

Message and Abend Code Explanation user exit. This exit can be used to provide message and abend code explanations to complement or substitute those provided by Fault Analyzer. For details, see "Message and Abend Code Explanation user exit" on page 377.

### FORMAT

Formatting user exit. This exit can be used to add user-specific information to the analysis report. For details, see "Formatting user exit" on page 381.

**END** End Processing user exit. This exit can be used to request suppression of the MVS system dump, the Fault Analyzer minidump, or the entire history file entry. For details, see "End Processing user exit" on page 393.

### NOTIFY

Notification user exit. This exit can, for example, be used to provide installation-specific notification of recorded faults. For details, see "Notification user exit" on page 397.

The exit name that is specified as *exit\_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

### NONE

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

### -DROPCNF-

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNF00 parmlib member. For details, see "Dropping IDICNF00 parmlib member user exit specifications" on page 475.

If one or more exits have been specified via the Exits option, information about the exits is written to the "Options in Effect" section of the analysis report. In this section, all specified exits are listed, and those of each type that were invoked (if any) are identified.

**Note:** The invocation status of the Notification user exit is not available as this exit is invoked after the completion of the analysis report.

An example of the information written to the "Options in Effect" section of the analysis report follows:

Exits:

The following user exits were specified via Exits options.

Type	Name	Type	Invoked
-----			
CONTROL	CTLEXITA	LMOD	No - module not found
	CTLEXITB	REXX	Yes
	CTLEXITC	REXX	Yes
	CTLEXITD	LMOD	No - module not found
END	ABC1	LMOD	Yes
NOTIFY	NONE	n/a	(Not attempted)

This example indicates:

- Four Analysis Control user exits had been specified. The first of these (CTLEXITA) was a load module that could not be invoked. The second user exit specified (CTLEXITB) was a REXX EXEC that was invoked. No attempt was

made to invoke the third (CTLEXITC) or fourth (CTLEXITD) user exits because of the successful invocation of the second.

- A single End Processing user exit (ABC1) had been specified as a load module. This user exit was invoked.
- A 'null' Notification user exit had been specified, which is never invoked.

## Dropping IDICNF00 parmlib member user exit specifications

**Note:** Information in this section is applicable to both the Exits and the DumpRegistrationExits options.

It is possible to drop all user exit specifications for a given exit type specified in the IDICNF00 parmlib member by using the special exit name, -DROPCNF-.

If, for example, the IDICNF00 parmlib member contained:

```
Exits(Control(FRED1,FRED2))
```

and an IDIOPTS user-options file contained:

```
Exits(Control(FRED3,-DROPCNF-,FRED4))
```

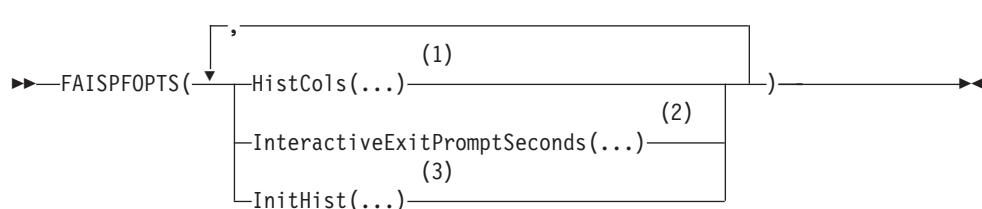
then the final logical concatenation of Analysis Control user exits would be:

Type	Name
<hr/>	
CONTROL	FRED3
	FRED4

Note that the -DROPCNF- exit name can be specified anywhere that an Exits option can be specified, with the exception of the IDICNF00 parmlib member.

## FAISPFopts

### Syntax



### Notes:

- 1 See "Histcols" on page 476.
- 2 See "InteractiveExitPromptSeconds" on page 477.
- 3 See "InitHist" on page 477.

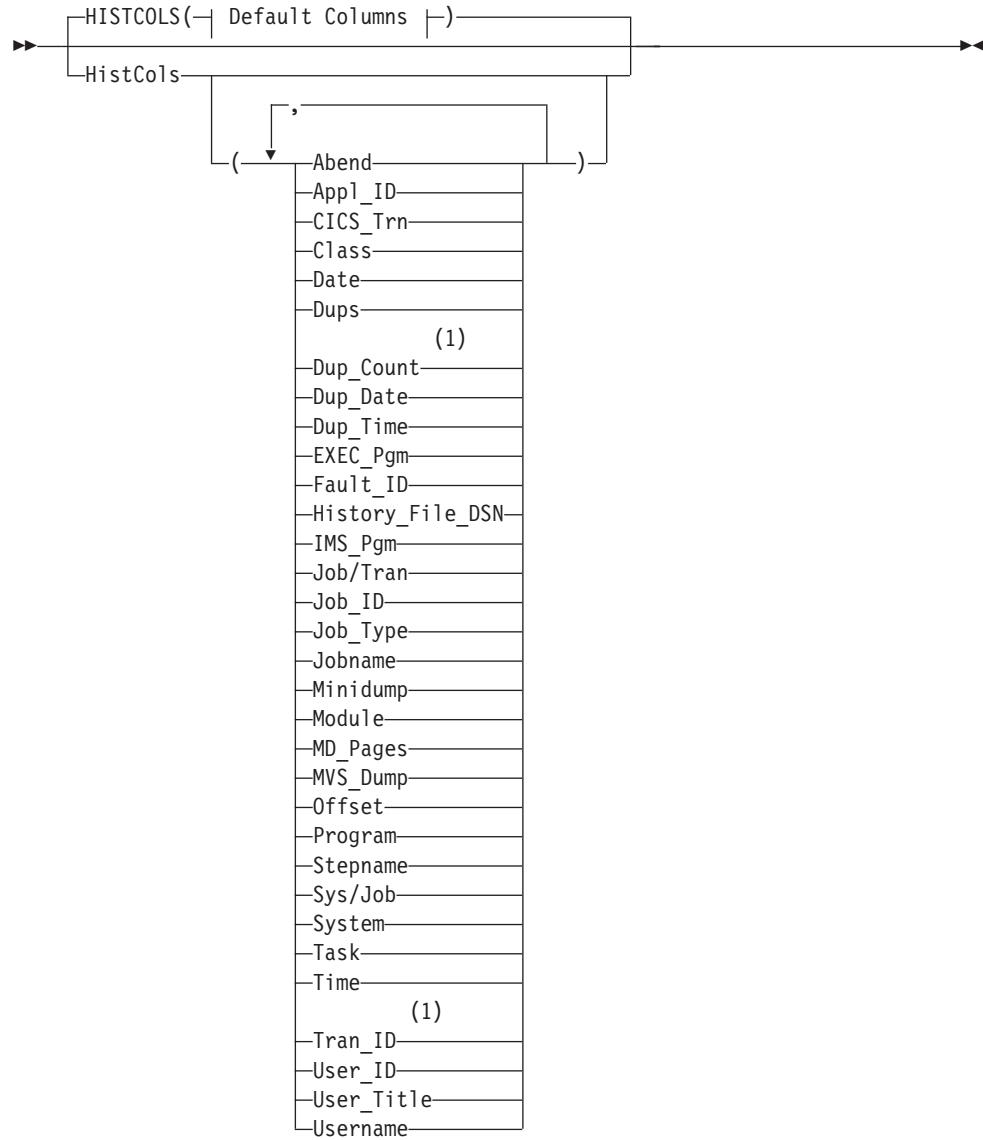
The FAISPFopts option is used to specify options for the Fault Analyzer ISPF interface.

This option is not included in the section of the fault analysis report that shows options in effect.

## HistCols

### HistCols

#### Syntax



#### Default Columns:

|—Fault\_ID—Job/Tran—User\_ID—Sys/Job—Abend—Date—Time—————|

#### Notes:

- 1    Deprecated column name—refer to “Fault entry list column configuration” on page 43 for details.

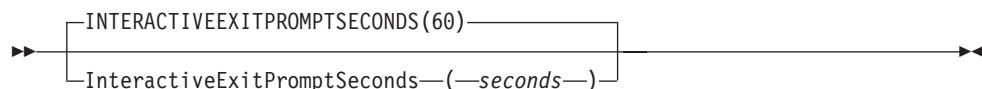
The Histcols option can be used to customize the columns of information that are shown on the Fault Entry List display when using the Fault Analyzer ISPF

interface. By adding this option to the IDICNF00 config member, an installation can provide all of its users with a base display that is different from the default display provided by Fault Analyzer. Users can change their own Fault Entry List display regardless of the specification of this option.

For more information about the customization of the Fault Entry List display, and for an explanation of the individual columns that can be specified using the HistCols option, see “Fault entry list column configuration” on page 43.

## InteractiveExitPromptSeconds

### Syntax



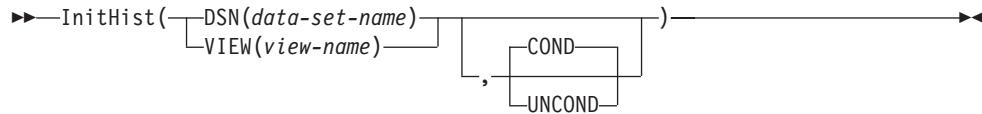
The InteractiveExitPromptSeconds option specifies the minimum number of elapsed seconds that the interactive reanalysis must have taken before the exit prompt panel is shown when leaving the interactive report display. If the interactive reanalysis took less than the specified number of seconds, then no prompt is shown. Otherwise, the user is requested to press the Enter key to confirm exit from the interactive report.

The valid *seconds* range is 0 - 99999:

- If 0 is specified, then the prompt is displayed unconditionally.
- If 99999 is specified, then the prompt is never displayed.

## InitHist

### Syntax



The InitHist option can be used to specify that a given history file or view should initially be displayed by all users of the Fault Analyzer ISPF interface.

### DSN(*data-set-name*)

Specifies the data set name of a history file that is to be used for the initial ISPF interface display.

### VIEW(*view-name*)

Specifies the name of a view member in the IDIVIEWS concatenation that is to be used for the initial ISPF interface display.

### COND

The specified history file or view is only used for the initial Fault Entry

## **InteractiveExitPromptSeconds**

List display shown when starting the Fault Analyzer ISPF interface, if the user has not previously selected a different history file or view.

This value is the default.

## **UNCOND**

The specified history file or view is always used for the initial Fault Entry List display shown when starting the Fault Analyzer ISPF interface, regardless of whether the user has previously selected a different history file or view.

## **FaultID**

### **Syntax**

```
►► [Faultid—(faultid)—] ◀◀
```

The FaultID option identifies a history file entry by its assigned fault identifier (*faultid*), and is used when performing fault reanalysis. The fault identifier must be in the format:

*cccnxxxx*

where *ccc* is the assigned history file prefix (1 to 3 characters) and *nnnnn* is a 5-digit decimal fault number.

#### **Note:**

1. The fault identifier F00000 signifies a 'null' ID that cannot be used to select a history file entry for reanalysis. This identifier might be assigned to a fault entry if no available fault numbers exist.
2. No default setting of the FaultID option is appropriate. It should only be used if building your own JCL for batch reanalysis of a history file fault entry. If performing reanalysis through the Fault Analyzer ISPF interface, then the correct use of this option is automatic.

## **GenerateSavedReport**

### **Syntax**

```
►► [NOGENERATESAVEDREPORT  
| GenerateSavedReport  
|   [Override  
|   | NoOverride  
| NOGenerateSavedReport] ] ◀◀
```

The GenerateSavedReport option can be used to create or replace a saved report in a fault entry during reanalysis.

If the Override suboption is used, then a new saved report is generated, regardless of whether one already exists or not.

If the NoOverride suboption is used, then a saved report is generated only if one does not already exist. This value is the default if the GenerateSavedReport option is specified without a suboption.

If the GenerateSavedReport option is in effect, then source information data sets (compiler listings or side files) used during real-time processing are automatically included during the reanalysis.

A practical application of this option might be to use it with automated batch reanalysis of newly created fault entries, which due to the DeferredReport option having been in effect during real-time processing for performance reasons, do not already include a saved report. By, for example, submitting such a batch reanalysis job from a Notification user exit, a saved report can be made to exist prior to users issuing the 'V' or 'S' line command from the Fault Entry List display. A sample Notification user exit which can be used for this purpose has been provided in "Example 4" on page 401.

If the GenerateSavedReport option is used with batch analysis of an MVS dump data set specified using the DumpDSN option, then this option causes a fault entry to be created in the current history file, subject to the user's access authorization. If the fault entry is to be created in a history file, other than the default history file, then the history file can be designated using any of the normal methods, such as specifying an IDIHIST DD statement or using the DataSets(IDIHIST(*dsn*)) option. If a fault entry is successfully created, then message IDI0164I is issued.

## **HistCols**

**Note:** This option has been deprecated and might be removed in a future release of Fault Analyzer. Instead, use the HistCols suboption of the FAISPFopts option ("HistCols" on page 476).

## **Include**

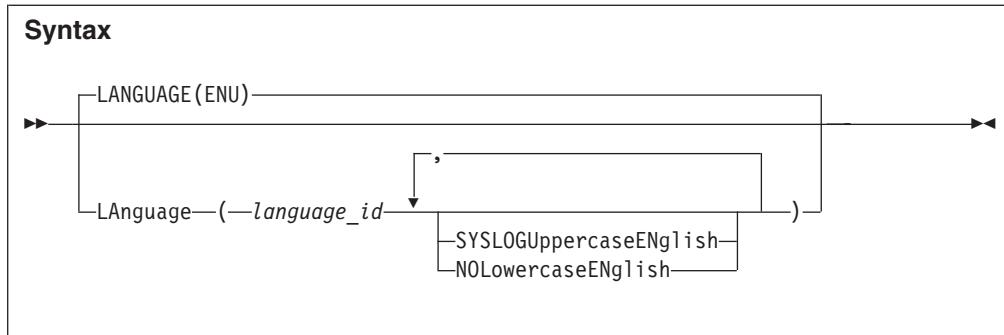
See "Exclude/Include" on page 469.

## **InteractiveExitPromptSeconds**

**Note:** This option has been deprecated and might be removed in a future release of Fault Analyzer. Instead, use the InteractiveExitPromptSeconds suboption of the FAISPFopts option ("InteractiveExitPromptSeconds" on page 477).

## **Language**

## Language



The Language option specifies the national language ID which is used to select appropriate language-dependant messages.

The following language IDs are permitted:

ID	Language
----	----------

ENU	U.S. English (default)
-----	------------------------

JPN	Japanese (available only if the Japanese feature of Fault Analyzer is installed)
-----	--

KOR	Korean (available only if the Korean feature of Fault Analyzer is installed)
-----	--

You can specify the following optional suboptions:

**SyslogUppercaseEnglish**

Where possible, causes all WTO messages to be in uppercase English only.

This suboption can be abbreviated to SYSLOGUEN.

**NoLowercaseEnglish**

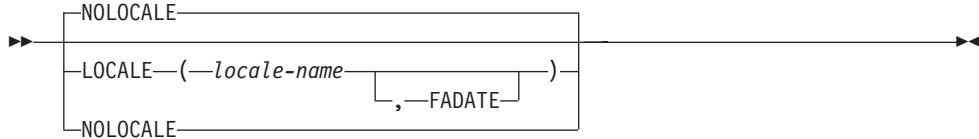
Where possible, folds English lowercase characters to uppercase in all output written by Fault Analyzer. This suboption can, for example, be used with 3270 terminals in 'KANA' mode to prevent lowercase English characters from being unreadable.

This suboption can be abbreviated to NOLEN.

If the Language option is specified in the IDICNF00 parmlib configuration member, then it should be the first option that is specified. This permits the maximum number of language-dependant messages to be issued by Fault Analyzer.

**Note:** If the Fault Analyzer IDIS subsystem is used, then any Language option specified by the user is ignored as far as explanations for abends, messages, reason codes, and similar, are concerned. These explanations are always provided in accordance with the Language option in effect for the IDIS subsystem. This is because the IDIS subsystem is used to cache explanations in order to improve performance and reduce I/O.

## Locale

**Syntax**

The Locale option specifies the locale to be used for cultural environment-dependent presentation. Affected are things like date and time formatting, collating sequences of sorted information, and determination of non-printable characters which are shown as periods.

The locale name that is specified as *locale-name* can be one of those supplied with z/OS C/C++ for the setlocale() runtime function. A list of locale names can be found in the *z/OS C/C++ Programming Guide*, "Appendix D. Locales Supplied with z/OS C/C++".

If the optional FADATE suboption is specified, then all dates presented are in the standard Fault Analyzer format of YYYY/MM/DD, regardless of the locale used.

Specifying the NoLocale option is the equivalent to specifying Locale(C,FADATE).

**Note:** The Fault Analyzer specification of *locale-name* overrides any IBM Problem Determination Tools for z/OS Common Component Locale option specification in the IPVCNF00 parmlib member. For information about the IPVCNF00 parmlib member, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

## LoopProtection

**Syntax**

This option is used to activate or deactivate the Fault Analyzer loop/wait protection feature. With this feature active (the default condition), Fault Analyzer terminates processing if the maximum expected elapsed time for any one of several internal checkpoints has been exceeded. If this termination happens, then message IDI0092S is issued.

A user exit can deactivate the loop/wait protection feature by setting the ENV.LOOPPROTECTION\_OPT field to N. For more information about this field, see "ENV - Common exit environment information" on page 513.

**Note:** Even with the NoLoopProtection option in effect, there is no guarantee that an analysis, that might otherwise be terminated with the IDI0092S message,

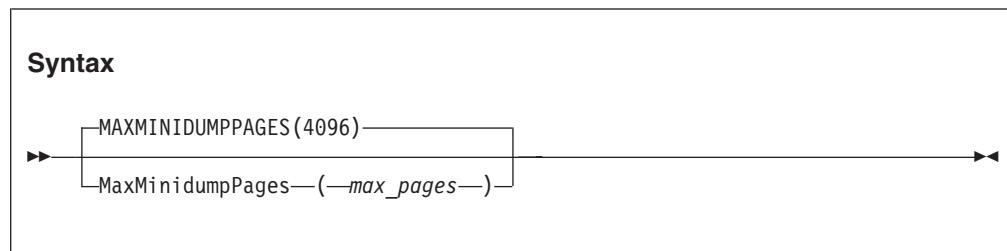
## LoopProtection

successfully completes. This possible lack of completion is because the job might still be subject to normal installation-imposed MVS execution time limits, or be in a never-ending loop or wait condition that eventually requires the job to be cancelled.

This option is only applicable to real-time analysis. For all other modes of execution, it is ignored.

This option is only included in the section of the fault analysis report that shows options in effect if real-time analysis with NoLoopProtection in effect.

## MaxMinidumpPages



The MaxMinidumpPages option specifies the maximum number of 4K pages that can be saved in the history file with each fault entry. Only pages that were referenced during the real-time analysis are saved as a 'minidump' against which reanalysis can be performed. If the number of referenced pages exceed *max\_pages*, then a fault entry with a saved report is still created, but it does not include a minidump.

**Note:** Additional minidump pages resulting from storage referenced during execution of a Formatting user exit are not included in the number of minidump pages tested against this option, nor are these included in the number of minidump pages provided as input to an End Processing user exit.

The MaxMinidumpPages option does not cause any preallocation of storage or DASD space. It is simply a cut-off mechanism to prevent minidumps above a specified size from being written with the history file fault entry.

This option does not apply to batch or interactive fault reanalysis.

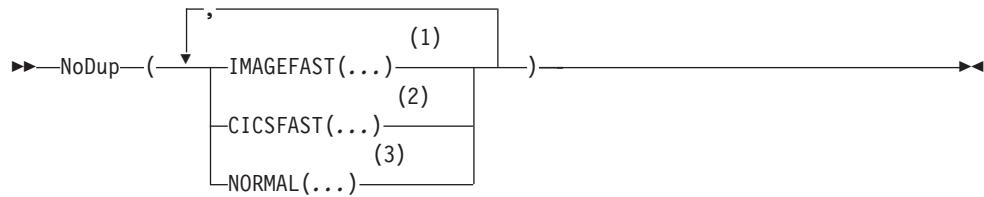
The section of the fault analysis report that shows options in effect includes this option and further indicate if the limit was exceeded, and if so, by how many pages. If it was exceeded, then message IDI0052I is also issued.

The valid *max\_pages* range is 0 - 524288.

**Note:** The use of an Analysis Control or End Processing user exit can effectively override the MaxMinidumpPages option in effect.

## NoDup

## Syntax



### Notes:

- 1 For details, see "NoDup(ImageFast(...))" on page 484.
- 2 For details, see "NoDup(CICSFAST(...))" on page 486.
- 3 For details, see "NoDup(NORMAL(...))" on page 489.

The NoDup option specifies duplicate fault detection thresholds.

There are two distinctly different types of duplicate fault detection, depending on the execution environment:

- **Fast duplicate detection:**

Applicable to CICS or IMS:

- CICS fast duplicate detection is controlled by the CICSFAST suboption of the NoDup option (see "NoDup(CICSFAST(...))" on page 486).  
The scope of duplicate detection is limited to a single CICS region.
- IMS fast duplicate detection is controlled by the ImageFast(IMS) suboptions of the NoDup option (see "NoDup(ImageFast(...))" on page 484).  
The scope of duplicate detection is across the entire MVS image.

The purpose of fast duplicate detection is mainly to prevent multiple identical abends, which all occur within a relatively short period of time, from unnecessarily slowing down abend recovery by only permitting one fault analysis of each unique problem within the specified period. Since the detection of duplicate faults must occur prior to fault analysis, the criteria differs from that used for normal duplicate detection.

Faults that have not been deemed duplicates based on the "fast" duplicate detection rules are subsequently subject to "normal" duplicate detection.

- **Normal duplicate detection:**

Applicable to all execution environments,

Controlled by the NORMAL suboption of the NoDup option (see "NoDup(NORMAL(...))" on page 489).

The purpose of normal duplicate detection is to prevent multiple identical faults from being recorded separately in a history file, thus assisting with DASD space preservation as well as maintaining a history file containing only entries which represent unique problems.

Regardless of the method used to designate a fault as a duplicate, the fault entry of which it is a duplicate has its duplicate count incremented accordingly.

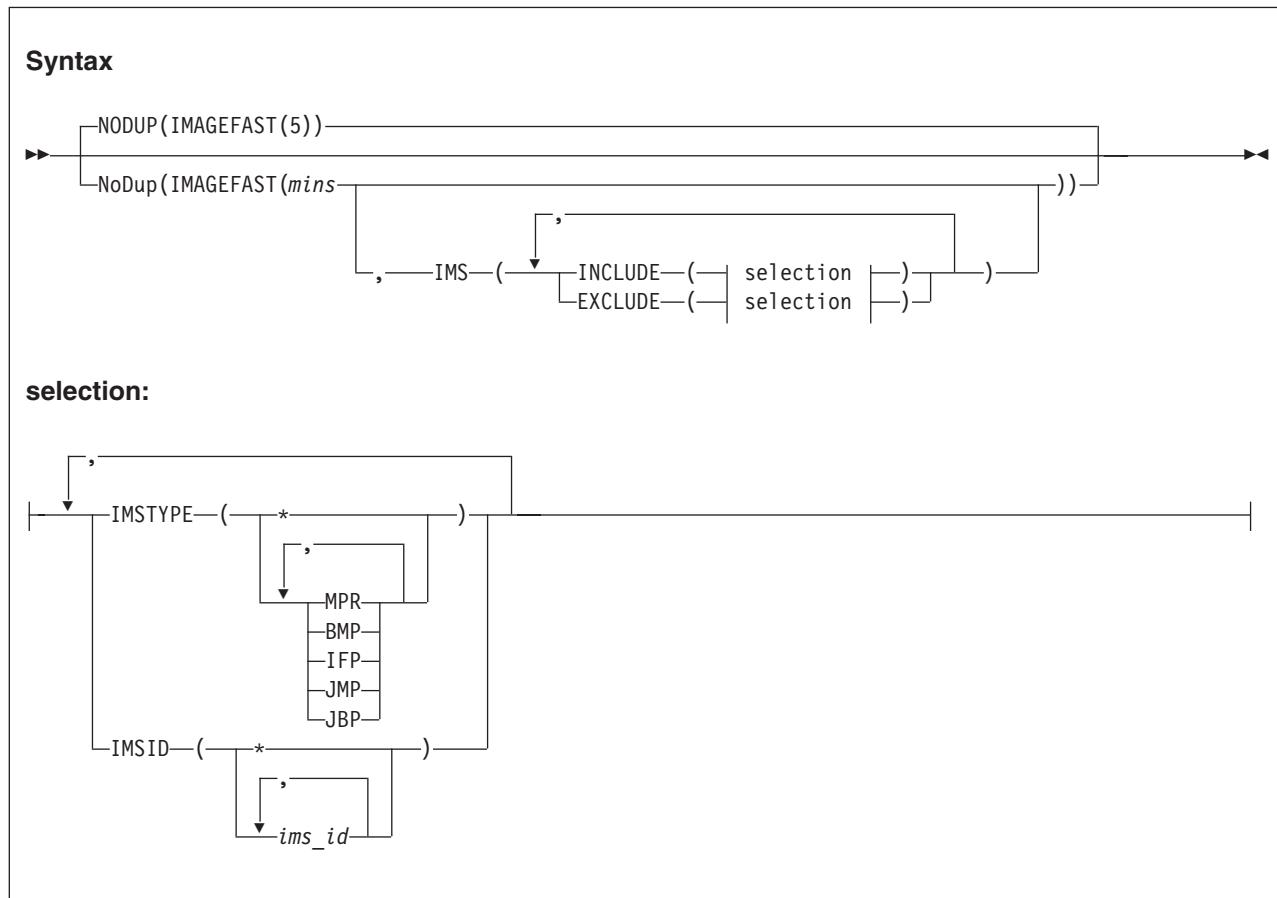
This option applies to the real-time analysis only.

## NoDup

If NoDup(NORMAL(*hours*, ...)) is in effect, with a non-zero value of *hours*, then this option is included in the section of the fault analysis report that shows options in effect. In addition, information about whether no duplicate fault was determined, or the fault ID of any identified duplicate, is provided. NoDup(CICSFAST) or NoDup(ImageFast) option information is not included, since no report is written if the associated duplicate criteria matched.

While the syntax for each duplicate subtype is shown separately in the following, they can be specified together in a single NoDup option if desired.

### NoDup(ImageFast(...))



This suboption is used to specify the number of minutes elapse time, since the last occurrence of a fault, during which subsequent invocations of Fault Analyzer for other faults *in the same MVS image* are deemed duplicates of the last fault, if they satisfy the appropriate fault characteristics criteria. This type of fault suppression is referred to as "IMS fast duplicate fault suppression".

**Note:** In order to enable IMS fast duplicate fault suppression, the Fault AnalyzerIDIS subsystem must be started with the IMAGEFAST PARM field option, as well as the UPDINDEX option. For details, see Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239.

Only PDSE history files that are managed by the IDIS subsystem are able to participate in the IMS fast duplicate fault suppression.

The valid range of the minutes value specified in *mins* is 0 - 10080 (10080 is equivalent to one week). Specification of 0 minutes means that "IMAGEFAST" duplicate faults are not detected.

The default elapse time is 5 minutes.

In the IMS environment, a fault is considered a duplicate of another if the faults occurred within the specified elapse time (*mins*), and the following fault details are identical:

- IMS program name
- IMS subsystem ID
- Last IMS status code
- Last DB2 SQLCODE
- Failing program name
- Failing program compile date
- Offset to error in failing program
- Length of failing program
- Abend code
- User title specified for IDISNAP invocation
- Call chain

NoDup(ImageFast) signatures are kept in the IDIS subsystem on each MVS image.

By default, all IMS jobs are eligible for IMS fast duplicate fault suppression. This approach is the equivalent of having specified the option

```
NoDup(ImageFast(minutes,IMS(INCLUDE(IMSTYPE(*),IMSID(*)))))
```

Use the INCLUDE or EXCLUDE suboptions of the NoDup(ImageFast(*minutes*,IMS(...))) option to restrict eligibility for IMS fast duplicate fault suppression. Here are the specification rules:

- An IMSTYPE criterion matches if the specified values include the IMS region type that is associated with the current fault.

For example, if the following option is specified:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR,BMP,IFP)))))
```

and the current fault is an IMS BMP region, then the IMSTYPE criterion matches, and the current fault deemed ineligible for IMS fast duplicate fault suppression.

- An IMSID criterion matches if one or more of the specified values for *ims\_id* match the IMS ID associated with the current fault.

For example, if the following option is specified:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSID(ABC1,ABC2,ABC3)))))
```

and the current fault is associated with the IMS subsystem ID ABC2, then the IMSID criterion matches, and the current fault deemed ineligible for IMS fast duplicate fault suppression.

- Each specification of an INCLUDE or EXCLUDE suboption is tested against the current fault and, if all criteria of the specified suboption match, changes the eligibility state accordingly.

For example, if the following option is specified:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR,BMP),IMSID(ABC1,ABC2,ABC3)))))
```

## NoDup

and the current fault is an IMS BMP region with subsystem ID ABC4, then the IMSTYPE criterion matches, but the IMSID criterion does not, resulting in the EXCLUDE criteria not matching.

Conceptionally, IMSTYPE or IMSID values can be considered logically OR'ed, while INCLUDE or EXCLUDE suboptions can be considered logically AND'ed, in order to determine the resulting match status. If | represents a "logical OR" operation, and & represents a "logical AND" operation, then the previous option specification could be interpreted as

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR | BMP) & IMSID(ABC1 | ABC2 | ABC3)))))
```

A single INCLUDE or EXCLUDE suboption should only ever contain a single IMSTYPE and/or a single IMSID criterion, since a match is otherwise not possible.

- Any number of INCLUDE or EXCLUDE suboptions can be specified. Specifying multiple INCLUDE or EXCLUDE suboptions within a single NoDup(ImageFast(*minutes*,IMS(...))) option is equivalent to specifying these as separate options.

For example, specifying

```
NoDup(ImageFast(5,IMS(INCLUDE(IMSTYPE(MPR),IMSID(ABC1)))))
```

is equivalent to specifying

```
NoDup(ImageFast(5,IMS(INCLUDE(IMSTYPE(MPR))))  
NoDup(ImageFast(5,IMS(INCLUDE(IMSID(ABC1))))))
```

**Note:** If different, only the last specified value of *mins* takes effect.

- A wildcard (\*) can be used as complete substitution for a value, that is, with no characters preceding the asterisk.
- Processing of INCLUDE or EXCLUDE suboptions occur in the order specified, and in accordance with the hierarchy of options sources defined in Chapter 33, "Options," on page 451. For example, the IDICNF00 parmlib member is read prior to any IDIOPTS user options file that was used.
- The most generic criteria should be specified first, followed by more specific ones.

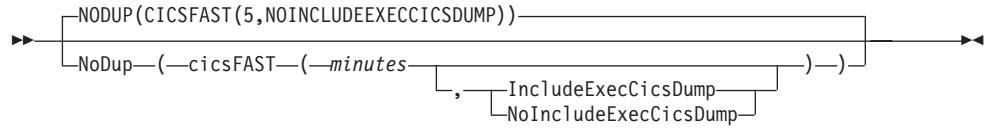
For example, if an installation wants to use IMS fast duplicate fault suppression for only IMS MPR jobs using subsystem IDs other than ABC1, then specifying the following options would achieve this:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(*) /* Exclude everything */  
INCLUDE(IMSTYPE(MPR)) /* Include only IMS MPR regions */  
EXCLUDE(IMSID(ABC1)) /* Exclude subsystem ID ABC1 */  
)))
```

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(ImageFast(...)) option in effect, the fault analysis is skipped, the duplicate count associated with the original fault is incremented by one, and message IDI0121I is issued.

**Changing the NoDup(ImageFast) option:** The NoDup(ImageFast) option is used by the Fault Analyzer IDIS subsystem only. Changes to the NoDup(ImageFast) option only take effect after stopping and restarting the Fault Analyzer IDIS subsystem (for details, see Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239).

## NoDup(CICSFAST(...))

**Syntax**

This suboption is used to specify the number of minutes elapse time, since the last occurrence of a fault, during which subsequent invocations of Fault Analyzer for other faults *in the same job step* are deemed duplicates of the last fault if they satisfy the appropriate fault characteristics criteria.

An illustration of this type of processing is shown in Figure 177 on page 488, where:

- All instances of “fault A” are considered duplicates of each other based on fault characteristics alone
- All instances of “fault B” are considered duplicates of each other based on fault characteristics alone
- The “fault A” characteristics do not match those of “fault B”

## NoDup

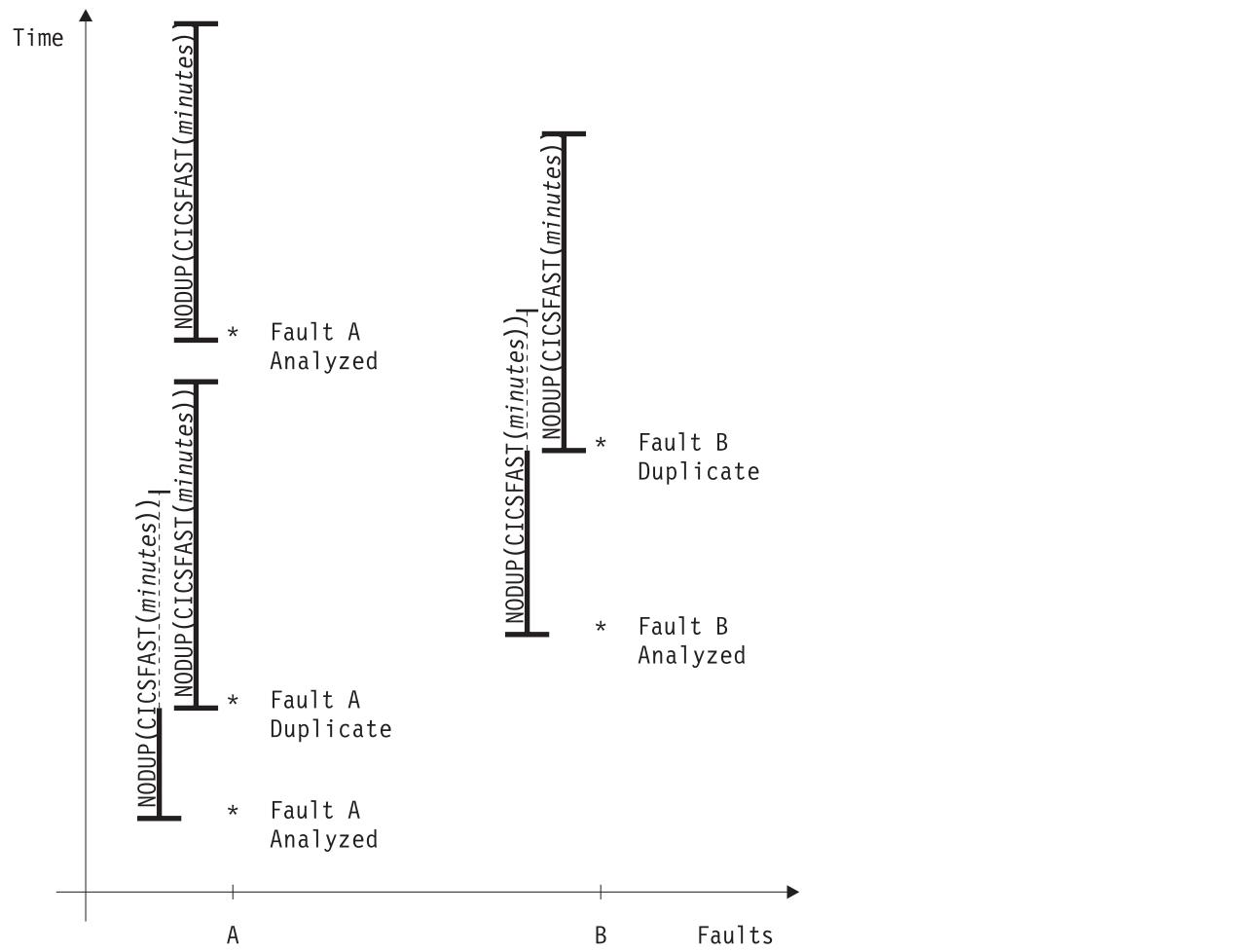


Figure 177. NoDup(CICSFAST) illustration

This option is currently used under CICS to prevent multiple identical transaction abends that occur within a short period of time from all being analyzed by Fault Analyzer with subsequent risk of exhausting system resources.

The valid range of *minutes* is 0 - 10080 (10080 is equivalent to one week). Specification of 0 minutes means that "CICSFAST" duplicate faults are not detected.

The default elapse time is 5 minutes.

In the CICS environment, a fault is considered a duplicate of another if the faults occurred within the specified elapse time (*minutes*), and the following fault details are identical:

- Transaction IDs
- CICS abend codes
- Failing program names
- Request IDs
- System and user sense codes
- Operating system abend codes
- Offsets to the point of error
- PSWs on entry to abend

NoDup(CICSFAST) signatures are kept in the CICS region.

All fault details are obtained from the transaction abend control block (TACB), except for the transaction IDs. Only TACB fields that are valid for both TACBs are included in the duplicate comparison.

By default, invocations of Fault Analyzer using the EXEC CICS DUMP command are not subject to NoDup(CICSFAST) duplicate determination. However, if the IncludeExecCicsDump suboption is specified, then NoDup(CICSFAST) duplicate determination is also performed when using the EXEC CICS DUMP command:

- If a TACB is available, then the duplicate determination is performed on the same basis as for CICS abends shown above (with the CICS dump codes used instead of the CICS abend codes).
- If a TACB is not available, then the duplicate determination is performed on the basis of:
  - Transaction IDs
  - CICS dump codes
  - Failing program names

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(CICSFAST(...)) option in effect, the writing of the history file entry is suppressed, the duplicate count associated with the last fault is incremented by one, and message IDI0066I is issued. This type of fault suppression is referred to as “CICS fast duplicate fault suppression”.

Fault Analyzer provides an exit, IDINDFUE, which can be used to override the duplicate designation of certain faults, based on abend code and other attributes. For details, see “CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)” on page 310.

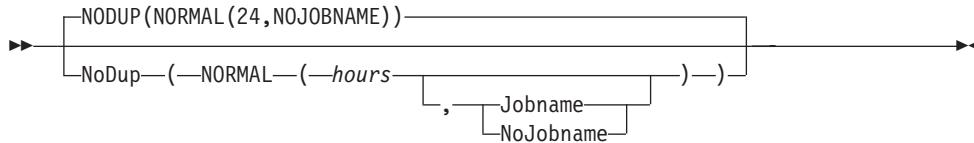
To prevent Fault Analyzer from continuing to designate new occurrences of abends as duplicates of an already deleted fault entry, the NoDup(CICSFAST) recording area should be cleared—for details, see “Clearing the NoDup(CICSFAST(...)) recording area” on page 307.

**Changing the NoDup(CICSFAST) option:** Typically, an IDIOPTS DD statement is used in CICS procedures to facilitate changes to options without the need to cycle CICS. However, as far as the NoDup(CICSFAST) option is concerned, a limitation exists. The actual processing of the IDIOPTS data set is performed by the main Fault Analyzer module, IDIDA. This module is the main program that runs in the MVS attached subtask. The NoDup(CICSFAST) processing is done in the IDIXCX52/IDIXCX53 exit code before the subtask is attached. The value used for NoDup(CICSFAST) is whatever the value was set to on the previous full run of Fault Analyzer in that CICS region (IDIDA attach). When a CICS fast duplicate fault suppression occurs (message IDI0066I), the attach of IDIDA and the reading of the options data set does *not* occur. Another IDIDA attach must happen before the NoDup(CICSFAST) option change is reflected down to the exit for NoDup(CICSFAST) action. This means either waiting for another unique fault (forced by creating a new abend with CECI to perform an EXEC CICS ABEND) or the NoDup(CICSFAST) time-out to occur.

## NoDup(NORMAL(...))

## NoDup

### Syntax



This suboption is used to specify the number of hours elapse time, inside of which invocations of Fault Analyzer, *using the same history file*, can be deemed duplicates of a previously recorded fault. This mechanism is the standard duplicate fault detection mechanism used, for example, for abending batch jobs.

The valid range of *hours* is 0 - 168 (168 is equivalent to one week). Specification of 0 hours means that "NORMAL" duplicate faults are not detected.

The default elapse time is 24 hours.

A fault is considered a duplicate of another already recorded in the same history file if the faults occurred within the specified elapse time (*hours*), the point-of-failure has been determined, and the following fault details are identical:

- Initial abend codes
- CICS transaction IDs (applicable to CICS transaction faults only)
- Point-of-failure module names
- Point-of-failure CSECT (or program) names
- Point-of-failure offsets from start of the CSECT (or program)
- Point-of-failure module link-edit date and time (if available for the current fault as well as in the history file entry for the duplicate fault)
- User titles.
- The jobname (only if the Jobname suboption is specified—otherwise, the jobname is not included in the duplicate fault determination)

NoDup(NORMAL) signatures are based on the existing fault entries in a history file.

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(NORMAL(*hours*)) option in effect, the default is to suppress both the writing of the history file entry and any dumps for the duplicate fault. However, the final decision on suppression can be overridden by an End Processing user exit.

When it has been determined that the fault is a duplicate of another fault, the duplicate count that is associated with the existing fault is incremented by one and message IDI0044I is issued.

**Trace information:** By adding an IDITRACE DDname to your job, Fault Analyzer can provide you with trace information that might help you understand why a particular fault is, or is not, being considered a "normal" duplicate of another fault during real-time analysis. For example:

```
//IDITRACE DD SYSOUT=*
```

(See "IDITRACE under CICS" on page 307 for an alternative method of activating this trace under CICS.)

**Note:** Only NoDup(NORMAL) trace information is available, not NoDup(CICSFAST).

The trace information is written to the IDITRACE DDname destination. An example of a NoDup option-processing trace follows:

---

```
NoDup(NORMAL(24)) option processing match values:
  Abend Code . . . . . : SOC7
  CICS Transaction ID. . . . . : n/a
  Module Name. . . . . . . . . : IDISCBL1
  CSECT Name . . . . . . . . . : IDISCBL1
  Offset . . . . . . . . . . . : X'3D4'
  Module Link-Edit Date. . . . . : 2002/05/06
  Module Link-Edit Time. . . . . : 15:56:35
Fault ID F00615 module link-edit time 15:55:09 did not match
No duplicate fault exists
```

---

Figure 178. Sample NoDup option-processing trace

Information about the reason for not matching (as for fault ID F00615 in the above example) is only provided for fault entries that are within the time interval in effect for the NoDup option, and if either of the following is true:

- The link-edit date did not match
- The link-edit time did not match
- The offset did not match

## PDTCCopts

### Syntax

```
►►PDTCCopts—(--CONFIGDSN—(--data-set-name--)--►►
```

The PDTCCopts option is used to specify Fault Analyzer settings for the IBM Problem Determination Tools for z/OS Common Component that are used by the Fault Analyzer web browser interface (for details, see “Using the Fault Analyzer web interface” on page 206).

#### CONFIGDSN(*data-set-name*)

Specifies the name of a data set (and optionally, member name in parenthesis) containing ISPF configuration settings. Although only the last two options, ISPF\_PROF\_DSN and ISPF\_APPL are required for the Fault Analyzer web browser interface, then this configuration file can be the same configuration file as the one that is also used for the Fault Analyzer plug-in for Eclipse. For details, see “Customize the IBM Problem Determination Tools for z/OS Common Component Common Server” on page 443.

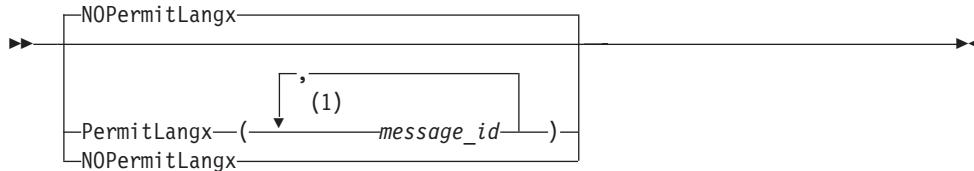
Sample specification:

```
CONFIGDSN(USER.PARMLIB(IDISRV03))
```

## PermitLangx

## PermitLangx

### Syntax



### Notes:

- 1 Either comma or blank character is permitted as delimiter.

Some compilers issue messages during the compilation which result in a return code greater than 4, while still producing a valid object module. However, since IDILANGX processing by default treats all messages that cause a return code greater than 4 as an error, this option can be used to specify a list of message IDs which should be ignored when found in the listing from a compilation ending with a return code greater than 4.

The PermitLangx option is applicable to COBOL or PL/I compiler listings only.

PL/I message IDs must be specified with a length of 8 characters. For example, if the following message was written to the compiler listing

Message	Statement	Message Description
IBM1352I E	11	The statement element PUT is invalid. The statement will be ignored.

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed

```
PermitLangx(IBM1352I)
```

OS/VS COBOL message IDs must be specified with a length of 8 characters. For example, if the following message was written to the compiler listing

CARD	ERROR MESSAGE
20	IKF1150I-C      ILLEGAL CHARACTER IN COLUMN 7, BLANK ASSUMED.

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed

```
PermitLangx(IKF1150I)
```

All other COBOL message IDs (other than OS/VS COBOL) must be specified with a length of 9 characters. For example, if the following message was written to the compiler listing

LineID	Message code	Message text
1	IGYDS0017-E	"IDENTIFICATION" should begin in area "A". It was processed as if found in area "A".

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed

```
PermitLangx(IGYDS0017)
```

## Using wildcards

To specify that all messages of a given severity should be permitted, the following notation can be used:

*XXX-C*

where

*XXX* The message prefix.

*C* The message severity level.

The following example shows how all error-level messages (E) can be permitted with Enterprise PL/I (IBM):

PermitLangx(IBM-E)

The message severity to which the specified option applies, is the one that follows the message ID in the compiler listing. For example, in the case of Enterprise PL/I message IBM1352I, the compiler listing might show:

Message	Statement	Message Description
IBM1352I	E	11 The statement element SKIP is invalid. The statement will be ignored.

In this case, the message severity level is E, which generally implies a return code of 8.

Only messages with the specified message severity level are affected.

The following example shows how to specify that both Enterprise PL/I and Enterprise COBOL error-level messages should be ignored:

PermitLangx(IBM-E,IGY-E)

## Combining multiple option specifications

Only the last specification of the PermitLangx option is used. That is, multiple specifications are not cumulative. For example, if the following options are specified in the IDICNFxx parmlib member

PermitLangx(IBM-E)  
 PermitLangx(IGYDS0017,IBM1352I)  
 PermitLangx(IGY-E)

then only the option

PermitLangx(IGY-E)

remains in effect.

To retain all three options that are specified, combine all suboptions into a single option specification, for example:

PermitLangx(IBM-E,  
 IGYDS0017,IBM1352I,  
 IGY-E)

## Maximum length of suboptions

The "normalized length", which is defined as the sum of the lengths of all message IDs, plus the number of message IDs minus one, must not exceed 75. This length is equivalent to the most compact specification possible, with only a single delimiter separating suboptions and no unnecessary blank-padding used. For example, the normalized length of

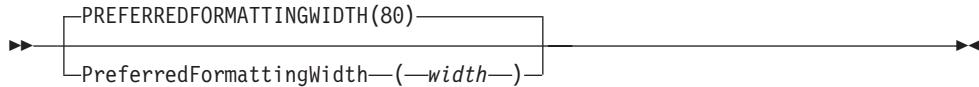
## PermitLangx

```
PermitLangx(IKF1150I      IBM-E,IGY-E    )
```

is 20. This value is calculated as follows: The total number of characters in the message IDs is  $8 + 5 + 5 = 18$ . Since 3 message IDs were specified, the total "normalized length" becomes  $18 + 3 - 1 = 20$ .

## PreferredFormattingWidth

### Syntax



This option specifies the real-time or batch reanalysis report line width to be used for flowing paragraph text and hexadecimal storage formatting. Other parts of the report are not affected by this option.

Hexadecimal storage formatting requires a minimum formatting width of 128 characters before changing from the default 16 bytes per line to 32 bytes per line.

The valid range of *width* is 80 - 132.

This option is not applicable to interactive reanalysis.

## PrintInactiveCOBOL

### Syntax



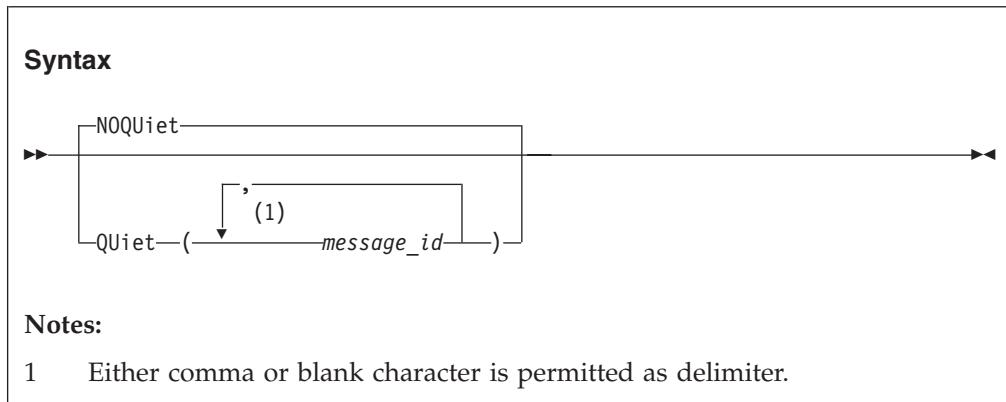
The PrintInactiveCOBOL option can be used with real-time or batch reanalysis to request that storage for inactive COBOL programs (programs that are not in the current save-area chain) is included in the analysis report.

**Note:** Depending on the number and size of inactive COBOL programs, this option might increase the size of the report significantly.

When the NoPrintInactiveCOBOL option is in effect (the default), then storage for inactive COBOL programs is not included.

This option is not applicable to interactive reanalysis, which is always capable of displaying storage for inactive COBOL programs.

## Quiet



To suppress specific write-to-operator messages, these can be specified as suboptions to the Quiet option. The message IDs must be the complete 8-character message ID, for example, IDI0003I.

If NoQuiet is specified, all write-to-operator messages can be issued.

The last specification of the Quiet or NoQuiet option is in effect. For example, if the IDICNF00 parmlib member specifies Quiet, this can be overridden by a user options file specification of NoQuiet. Each specification of a message list using the Quiet option overrides any previous specification completely, that is, the message numbers are not cumulative.

While this option can be used to reduce the number of messages written to the SYSLOG, its use is not generally recommended, since the lack of messages can make it difficult to diagnose problems with the Fault Analyzer processing. Also, certain diagnostic settings requested used by a Fault Analyzer service representative, while gathering documentation for a reported problem, might override this option.

Some messages are not able to be suppressed using the Quiet option, such as message IDI0001I.

It is not possible to suppress S-level (severe) messages, or messages written by the Fault Analyzer IDIS subsystem.

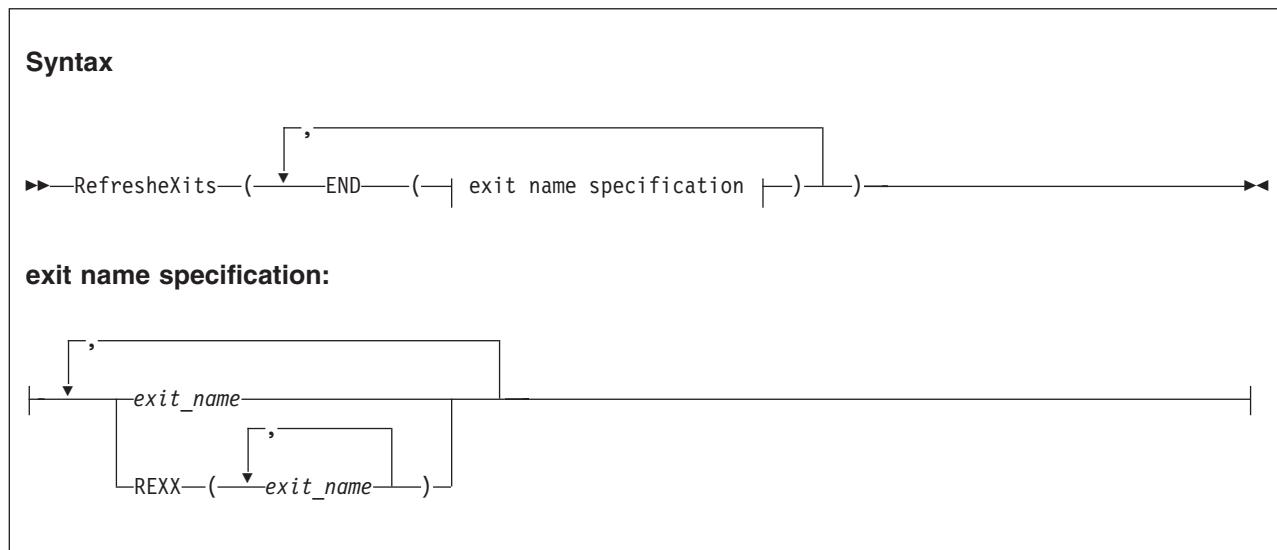
## RDZClient



**Note:** This option has been deprecated. It may still be specified, but is ignored.

## RefreshExits

### RefreshExits



The RefreshExits option specifies the types and names of user exits to be invoked during the first batch reanalysis of a dump registration fault entry. Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

Multiple specifications of the RefreshExits option are cumulative.

Exits can be either REXX EXECs or load modules:

- REXX EXECs must be specified as  
REXX(*exit\_name\_1*, *exit\_name\_2*, ...)
- and be available via the IDIEXEC DDname.
- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

**END** End Processing user exit. This exit can be used to request suppression of the Fault Analyzer minidump or the update of the entire history file entry. For details, see "End Processing user exit (Fault entry refresh)" on page 396.

The exit name that is specified as *exit\_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

**NONE**

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

**-DROPCNF-**

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNF00 parmlib member. For details, see "Dropping IDICNF00 parmlib member user exit specifications" on page 475.

If one or more exits have been specified via the RefreshExits option, information about the exits is written to the “Options in Effect” section of the analysis report. In this section, all specified exits are listed, and those of each type that were invoked (if any) are identified.

An example of the information written to the “Options in Effect” section of the analysis report follows:

Exits:

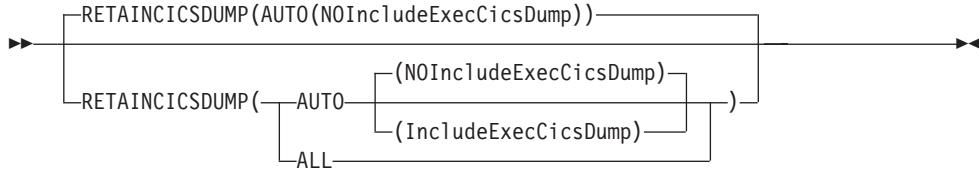
The following user exits were specified via RefreshExits options.

Type	Name	Type	Invoked
END	ABC1	LMOD	Yes

This example indicates that a single End Processing user exit (ABC1) had been specified as a load module. This user exit was invoked.

## RetainCICSDump

### Syntax



The RetainCICSDump option specifies that a CICS transaction dump should be retained based on the analysis success (AUTO) or unconditionally (ALL).

The analysis success (AUTO) setting means that if the Fault Analyzer analysis completes normally, then the transaction dump is suppressed. However, if there is a significant problem with the analysis, then the transaction dump is retained. AUTO(NOIncludeExecCicsDump) is used to prevent suppression of the CICS transaction dump issued for an EXEC CICS DUMP call, even if analysis is successful. This is the default. AUTO(IncludeExecCicsDump) is used to suppress all CICS transaction dumps, including for EXEC CICS DUMP calls, if the analysis is successful.

This option applies to real-time analysis of CICS transaction faults only, and requires that Fault Analyzer is installed in either the XPCABND or XDUREQ global user exits.

**Note:** Transaction dumps as a result of an EXEC CICS DUMP command are not suppressed by Fault Analyzer. This lack of suppression is due to the SUPPRESSED response being passed back to the issuing application program, which if not handled correctly, can lead to AEXW abends.

This option does not affect the writing of the fault entry to the history file.

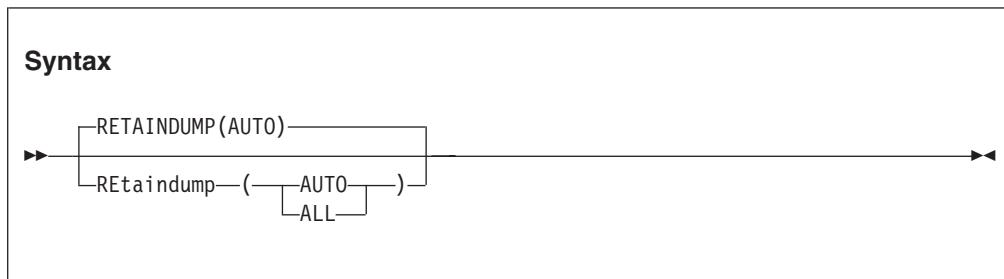
# RetainCICSDump

This option is included in the section of the fault analysis report that shows options in effect for real-time analysis of CICS transaction faults only.

## Note:

1. The use of an End Processing user exit can effectively override the RetainCICSDump option in effect.
  2. To control the retention of MVS dump data sets, use the RetainDump option instead.
  3. See “Dump suppression” on page 15 for more information about dump suppression.

# RetainDump



The RetainDump option specifies that the Fault Analyzer IEAVTABX change options/suppress dump exit (IDIXDCAP) is to retain the SYSABEND dump, SYSMDUMP, or SYSUDUMP of the abending job step, based on the analysis success (AUTO) or unconditionally (ALL).

The analysis success (AUTO) setting means that if the Fault Analyzer analysis completes normally, then the MVS dump is suppressed. However, if there is a significant problem with the analysis, then the MVS dump is retained.

This option applies to real-time analysis of non-CICS transaction faults only, and is only applicable to the IEAVTABX change options/suppress dump exit, IDIXDCAP. This option does not affect the writing of the fault entry to the history file.

If an exit is installed ahead of IDIXDCAP in the IEAVTABX exit list, and this exit has requested, via its return code, that dump options should be changed or the dump suppressed, then Fault Analyzer honors the request regardless of options settings.

This option is included in the section of the fault analysis report that shows options in effect for real-time analysis of non-CICS transaction faults only.

### Note:

1. The use of an Analysis Control or End Processing user exit can effectively override the RetainDump option in effect.
  2. To control the retention of CICS transaction dumps, use the RetainCICSDump option instead.
  3. See “Dump suppression” on page 15 for more information about dump suppression.

## Source

### Syntax



The Source option is used to control whether real-time source code analysis is to be performed:

- If Source is in effect, then real-time source code analysis is performed as usual. This value is the default.
- If NoSource is in effect, then Fault Analyzer does not access any compiler listings or side files and source code information might not be available in the real-time report. Assuming that either a minidump or a SYSMDUMP data set has been written, then it is still possible to provide source code information by performing reanalysis against the history file entry.

This option might be useful as a means of improving real-time analysis performance in installations where a large number of compiler listing or side file data sets are used, and where performing reanalysis to provide the extra source code information is an acceptable alternative.

## SpinIDIREPRT

### Syntax



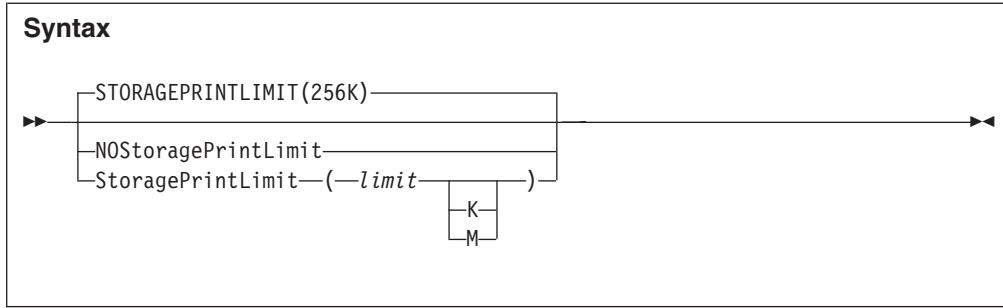
The NoSpinIDIREPRT option can be used with real-time analysis to request that dynamic allocation of the IDIREPRT DDname is not explicitly deallocated at the end of processing. Use this option, for example, to ensure that all JES spool data for a given real-time analysis is capable of being sent as a single file across to a different system using NJE.

When the SpinIDIREPRT option is in effect (the default), then dynamic allocation of the IDIREPRT DDname is explicitly deallocated at the end of processing.

This option is not applicable to batch or interactive reanalysis.

## StoragePrintLimit

## StoragePrintLimit



This option can be used to limit the size of reports when large amounts of virtual storage are eligible for display in the report, such as programs using large arrays in working storage.

The effect that this option has on the report is to determine the style of formatting used for event-related associated storage areas as follows:

*Table 13. StoragePrintLimit option behavior*

Compiler listing or side file available	StoragePrintLimit not exceeded	StoragePrintLimit exceeded
Yes	Complete event-related source view formatting	Addressable event-related source view formatting
No	Complete event-related hex-dump formatting	Addressable event-related hex-dump formatting

"Addressable" storage is storage that is within +4 kilobytes of a general purpose register value.

To determine if the specified limit is exceeded, it is tested against the sum of all hex-dumped event-related associated storage areas and CICS transaction storage areas being reported on, with due consideration given to overlapping storage ranges. The SystemWidePreferred(StorageAreas(Hex)) option is assumed in effect for the accumulation of storage ranges, regardless of its actual setting.

This option affects the real-time or batch reanalysis reports only—it does not affect the interactive reanalysis report. Even if used in real time, the full data is still available from reanalysis of the minidump, which is not reduced in size by the StoragePrintLimit option.

Valid specification of *limit*:

- Bytes: 0-2147483648
- Kilobytes: 0K-2097152K
- Megabytes: 0M-2048M

NoStoragePrintLimit is the equivalent to specifying StoragePrintLimit(2147483648).

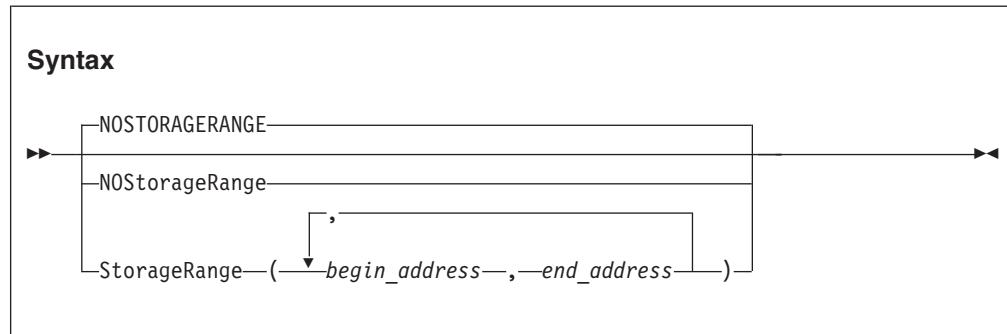
Information about the accumulated storage amount, which is used to determine whether or not the storage print limit in effect has been exceeded, is written to IDITRACE.

In the "Options in Effect" section of the report, this option is shown using the largest possible unit type regardless of the actual specification of the option. For example, if StoragePrintLimit(1024K) was specified, then the option is shown as

StoragePrintLimit(1M). If instead StoragePrintLimit(1025) was specified, then the option is shown as StoragePrintLimit(1025) also, since 1025 is not an even number of megabytes or kilobytes.

Information about whether the storage print limit was exceeded or not is included in the "Options in Effect" section of the report. If the limit was exceeded, then the amount of storage by which the limit was exceeded is also shown.

## **StorageRange**



This option can be used to request that one or more specific areas of storage are shown in the analysis report, instead of the default event-associated or system-wide hex-dumped storage.

The primary usage of this option is expected to be with IDISNAP calls from application programs, where only a certain area (or areas) of storage is of interest. For example, a COBOL programmer can choose to call IDISNAP with a StorageRange option that specifies a particular variable in the program's working-storage section. Instead of the analysis report containing all of the program's associated storage, it instead contains only the area requested.

For information about how to specify this option in a call to IDISNAP, see "Using the program SNAP interface (IDISNAP)" on page 20.

If this option is in effect for real-time analysis, then the same specification is in effect during any subsequent reanalysis of the fault entry, without the ability to override the option.

The begin address is the address of the first byte of the storage area, while the end address is the address of the byte immediately following the last byte of the storage area. That is, the end address minus the begin address equals the length of the storage area to be included.

Multiple storage ranges can be specified in any order, as long as the end address is greater than the begin address. Any invalid storage ranges are ignored.

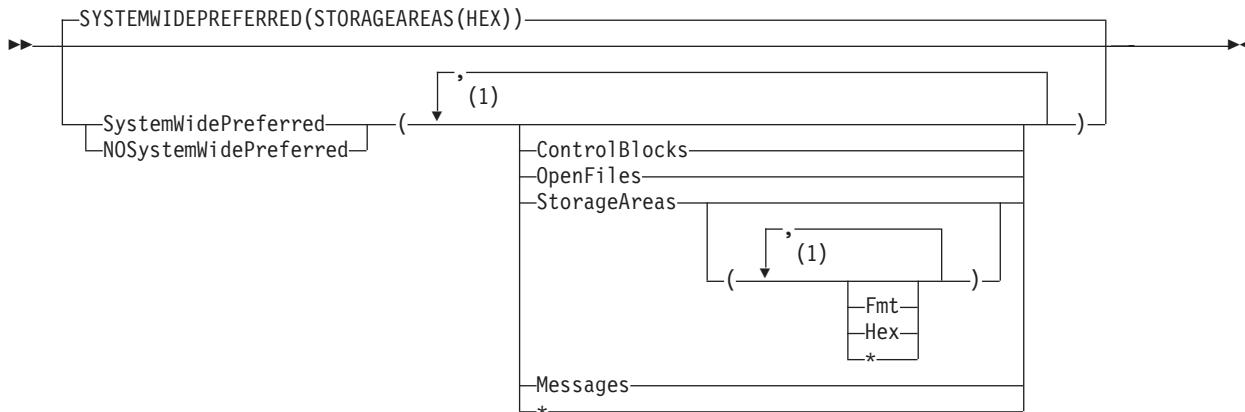
Multiple specifications of the StorageRange option replace any prior specifications, that is, the option is not cumulative.

The StorageRange option is shown in the "Options in Effect" section of the analysis report only if it has been specified in real time with at least one valid storage range.

## SystemWidePreferred

# SystemWidePreferred

## Syntax



## Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

This option is applicable to the real-time or batch reanalysis reports only. It does not affect the interactive reanalysis report.

Fault Analyzer by default places information that is generally considered associated with an event in the "Event Details" section for that event. However, by using the SystemWidePreferred option, Fault Analyzer can be directed to instead placing such information in the "System-Wide Information" section of the report.

The following suboptions are available to change the placement of various event-related areas of the report:

## ControlBlocks

Specifies that all event-related data that is usually placed under the heading "Associated Control Blocks" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead. This data includes the following control blocks for the various execution environments:

*Table 14. Control blocks affected*

<b>Execution environment</b>	<b>Control blocks affected</b>	<b>Target System-Wide Information section subheading</b>
DB2	SQLCA	"DB2 Control Blocks" in the "DB2 Information" section
IMS	AIB, DIB, and UIB	"IMS Control Blocks" in the "IMS Information" section

Explanations for any control block fields are placed following the control blocks in the "System-Wide Information" section of the report.

This suboption can be abbreviated to CB.

**OpenFiles**

Specifies that all event-related data that is usually placed under the heading "Associated Open Files" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead.

This suboption can be abbreviated to OF.

**StorageAreas**

Specifies that all event-related data that is usually placed under the heading "Associated Storage Areas" in the "Event Details" section of the report are instead to be placed in the "System-Wide Information" section under the subheading "Storage Areas".

The types of storage areas can further be selected using the following StorageAreas suboptions:

**Fmt** This suboption specifies high-level language program storage areas for which compiler listings or side files are available, and therefore are presented formatted in the report as opposed to hex-dumped. These areas are given separate subheadings within the "Storage Areas" section for easier identification.

This suboption can be abbreviated to F.

**Hex** This suboption specifies storage areas that are presented in hex-dump format. These types of storage areas are combined for all events and placed in ascending address sequence under the common subheading "Hex-Dumped Storage" within the "Storage Areas" section, with labeling of addresses to indicate their origin. This suboption is the default.

This suboption can be abbreviated to H.

\* Specifying an asterisk (\*) implies specification of all suboptions.

This suboption can be abbreviated to SA.

**Messages**

Specifies that all event-related messages that are usually placed under the heading "Associated Messages" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead.

This suboption can be abbreviated to M.

\* Specifying an asterisk (\*) implies specification of all suboptions.

Each specification of the SystemWidePreferred option is cumulative. To reset all SystemWidePreferred suboptions at once, use the NoSystemWidePreferred option.

**UseIDISTime****Syntax**

## **UseIDISTime**

The UseIDISTime option has been deprecated but is still permitted specified for compatibility with earlier levels of Fault Analyzer. The default is UseIDISTime, and although NoUseIDISTime can be specified, it is ignored.

Fault Analyzer obtains the current local time using STCK (STORE CLOCK), adjusted by CVTLDTO and CVTLSO. This time is normally not affected by clock simulators, such as IBM HourGlass, which can be used to modify the local time for application testing purposes.

---

## Chapter 34. Data areas

The following provides descriptions of data areas available to user exits. For information about user exits, refer to Chapter 31, “Customizing Fault Analyzer by using user exits,” on page 359.

Softcopy versions of the following data areas are available in the softcopy samples data set (IDI.SIDISAMP) for Assembler, COBOL, C, and PL/I as members IDISXPLA, IDISXPLB, IDISXPLC, and IDISXPLP respectively.

Notes relating to the following data areas:

- The 'Access' column provides information about a user exit's ability to update individual fields as follows:

**R/W** Read/write. These fields can be updated by the user exit.

**R/O** Read only.

- Numerical fields are identified by '(nnn)' and are, on input to the user exit, always padded on the left with zeroes to the total width of the field.

**Note:** Leading zeroes are generally not included in data area fields provided to REXX user exits.

- All fields for which Fault Analyzer does not provide an initial value are initialized to blanks.

**Note:** Blank-padding is generally not included in character-format data area fields provided to REXX user exits, unless the fields might contain hexadecimal characters.

- Unless otherwise indicated, all fields are translated to upper case by Fault Analyzer.
- Unless otherwise indicated, all R/W fields can be truncated using a null character (X'00') as delimiter. Truncation is never used by Fault Analyzer when initializing the parameter lists.
- For COBOL, substitute all underscores ('\_') with dashes ('-').

To see the initial values of any of these fields for an abending job, add the following JCL statements to produce an exit trace:

```
//IDITRACE DD SYSOUT=*
//IDIOPTS DD *
  Exits(exit_type(NONE))
/*
```

where *exit\_type* is either CONTROL, LISTING, FORMAT, REPORT, MSGXPL, END, or NOTIFY, depending on the data area that you are interested in. For more information about tracing, see “Diagnostic tracing” on page 363. For more information about the Exits option, see “Exits” on page 473.

---

### Non-REXX user exit buffered data format

The information that is provided in this section is only applicable to user exits that are not written in REXX.

## Non-REXX user exit buffered data format

If the data length for certain character fields exceed the maximum field size, then Fault Analyzer instead allocates a buffer large enough for all of the field data, and provide information about the buffer in three fullwords starting at relative offset zero of the field as follows:

- The first byte of the first fullword is set to X'FF' to indicate that this field contains buffer information. The remaining 3 bytes are not used, but are set to X'00' by Fault Analyzer.
- The second fullword is the address of a buffer containing the data in the same format as when the data is provided in the field itself.
- The third fullword is the allocated length of the buffer.

The data fields that this buffered format is applicable to are identified separately in the data area descriptions.

The Analysis Control user exit cannot free any buffer that is allocated by Fault Analyzer for the field. Instead, if the field or buffer size is inadequate, the exit can allocate its own buffer and place the address and length information in the three fullwords at relative offset zero of this field. Fault Analyzer is not dependent on any original buffer address to be retained for later release of allocated storage.

Freeing of buffers that were allocated by user exits is the responsibility of the user. This freeing can be achieved be utilizing the ENV.USER\_1 or ENV.USER\_2 fields to point to an area of storage containing information about any buffer allocations made. A later exit, such as the End Processing user exit, can then be used to perform the release of allocated storage.

For REXX user exits, Fault Analyzer automatically handles the allocation and freeing of any necessary buffers to accommodate data lengths in excess of the field size.

---

## Data area descriptions

The following describes the various user exit data areas.

### CTL - Analysis Control user exit parameter list

Table 15. CTL data area

Offsets		Type	Access	Len	Name and description
Dec	Hex				
0	(0)	CHAR	R/O	4	<b>VERSION</b> Parameter list version (currently 0002).
4	(4)	CHAR	R/W	1	<b>EXCLUDE</b> Exclude from analysis (Y/N). If this field is set to 'Y', no analysis of the current fault is performed, and no updates are made to the history file.  Only applicable to real-time processing and dump registration.

Table 15. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
5	(5)	CHAR	R/W	1	<b>DETAIL_OPT</b>
					Detail option (S/M/L). Refer to "Detail" on page 465 for information about this option.
					Not applicable to interactive reanalysis.
6	(6)	CHAR	R/W	1	<b>DEFERREDREPORT_OPT</b>
					DeferredReport option (Y/N). Refer to "DeferredReport" on page 463 for information about this option.
					Only applicable to real-time processing.
7	(7)	CHAR	R/W	4	<b>DETAIL_OPT_EXTRA_SOURCE</b>
					Number of extra source code lines or statements to be included in the real-time or batch reanalysis report ahead of, and after, the source line or statement that matches the CSECT offset (nnnn). The extra source lines or statements are included in the report detail section only.
					The default is 0005.
					Not applicable to interactive reanalysis.
8	(8)	CHAR	R/O	7	(Reserved)
18	(12)	CHAR	R/W	10	<b>RETAINDUMP_OPT</b>
					RetainDump option. Refer to "RetainDump" on page 498 for information about this option. (The format provided in this field is identical to that of the normal RetainDump option syntax, that is, "AUTO" or "ALL").
					Only applicable to real-time processing.
28	(1C)	CHAR	R/O	44	(Reserved)
72	(48)	CHAR	R/O	120	<b>INCLUDE_CRITERION</b>
					Include criterion. The last matching Include option criterion processed.
					Fault Analyzer permits a buffered data format that is used if the length of the criterion exceeds the maximum size that can be contained in this field. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
					Only applicable to real-time processing.

## CTL data area

Table 15. CTL data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
192	(C0)	CHAR	R/O	120 <b>EXCLUDE_CRITERION</b>  Exclude criterion. The last matching Exclude option criterion processed.
				Fault Analyzer permits a buffered data format that is used if the length of the criterion exceeds the maximum size that can be contained in this field. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
				Only applicable to real-time processing.
312	(138)	CHAR	R/O	5 (Reserved)
317	(13D)	CHAR	R/W	1 <b>SOURCE_OPT</b>  Source option (Y/N). Refer to "Source" on page 499 for information about this option.
				Only applicable to real-time processing.
318	(13E)	CHAR	R/O	6 (Reserved)
324	(144)	CHAR	R/W	1 <b>PRINTINACTIVECOBOL_OPT</b>  PrintInactiveCOBOL option (Y/N). Refer to "PrintInactiveCOBOL" on page 494 for information about this option.
				Not applicable to interactive reanalysis.
325	(145)	CHAR	R/O	51 (Reserved)
<b>Field format and usage:</b>				
• <u>The following information is applicable to all IDI*_PRE, IDI*_JOB, and IDI*_CFG fields in this data area:</u>				
In each field, the data set names are provided left justified and blank padded on a 45-character boundary. Any unused space is set to all blanks.				
Fault Analyzer permits a buffered data format that can be used if the number of data sets exceed the maximum number that can be contained in this field. The buffer, when allocated by Fault Analyzer, is allocated large enough to hold an extra 25 data set names. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.				
• <u>The following information is applicable to all IDI*_PRE fields in this data area:</u>				
An Analysis Control user exit can choose to unallocate any of these data sets, or allocate extra ones. These fields contain the preallocated data set names, and are provided for information only. Changes to the data set names in this field are not acted upon by Fault Analyzer. Instead, Fault Analyzer redetermines the preallocated data sets using standard operating system interfaces.				
• <u>The following information is applicable to all IDI*_JOB or IDI*_CFG fields in this data area:</u>				
An Analysis Control user exit can add to, delete, or alter any or all of these data set names. Upon return from the exit, data set names must be separated by one or more blanks or commas, and can be aligned on any boundary. For these data set names, Fault Analyzer uses the returned list of names.				
376	178	CHAR	R/O	5400 <b>IDIADATA_PRE</b>
				IDIADATA preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDIADATA DDname.

Table 15. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
5776	(1690)	CHAR	R/W	5400	<b>IDIADATA_JOB</b>  IDIADATA data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDIADATA data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
11176	(2BA8)	CHAR	R/W	5400	<b>IDIADATA_CFG</b>  IDIADATA data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDIADATA data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
16576	(40C0)	CHAR	R/O	5400	<b>IDILC_PRE</b>  IDILC preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILC DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
21976	(55D8)	CHAR	R/W	5400	<b>IDILC_JOB</b>  IDILC data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILC data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
27376	(6AF0)	CHAR	R/W	5400	<b>IDILC_CFG</b>  IDILC data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILC data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
32776	(8008)	CHAR	R/O	5400	<b>IDILCOB_PRE</b>  IDILCOB preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILCOB DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.

## CTL data area

Table 15. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
38176	(9520)	CHAR	R/W	5400	<b>IDILCOB_JOB</b>  IDILCOB data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILCOB data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
43576	(AA38)	CHAR	R/W	5400	<b>IDILCOB_CFG</b>  IDILCOB data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILCOB data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
48976	(BF50)	CHAR	R/O	5400	<b>IDILCOBO_PRE</b>  IDILCOBO preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILCOBO DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
54376	(D468)	CHAR	R/W	5400	<b>IDILCOBO_JOB</b>  IDILCOBO data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILCOBO data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
59776	(E980)	CHAR	R/W	5400	<b>IDILCOBO_CFG</b>  IDILCOBO data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILCOBO data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
65176	(FE98)	CHAR	R/O	5400	<b>IDILANGX_PRE</b>  IDILANGX preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILANGX DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.

Table 15. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
70576	(113B0)	CHAR	R/W	5400	<b>IDILANGX_JOB</b>  IDILANGX data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILANGX data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
75976	(128C8)	CHAR	R/W	5400	<b>IDILANGX_CFG</b>  IDILANGX data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILANGX data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
81376	(13DE0)	CHAR	R/O	5400	<b>IDILPLI_PRE</b>  IDILPLI preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILPLI DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
86776	(152F8)	CHAR	R/W	5400	<b>IDILPLI_JOB</b>  IDILPLI data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILPLI data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
92176	(16810)	CHAR	R/W	5400	<b>IDILPLI_CFG</b>  IDILPLI data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILPLI data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
97576	(17D28)	CHAR	R/O	1024	(Reserved)
98600	(18128)	CHAR	R/O	5400	<b>IDILPLIE_PRE</b>  IDILPLIE preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILPLIE DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.

## CTL data area

*Table 15. CTL data area (continued)*

Offsets		Type	Access	Len	Name and description
Dec	Hex				
104000	(19640)	CHAR	R/W	5400	<b>IDILPLIE_JOB</b>  IDILPLIE data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDILPLIE data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
109400	(1AB58)	CHAR	R/W	5400	<b>IDILPLIE_CFG</b>  IDILPLIE data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDILPLIE data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
114800	(1C070)	CHAR	R/W	256	<b>LOCALE</b>  Locale option locale name. Refer to "Locale" on page 480 for information about this option.
115056	(1C170)	CHAR	R/W	1	<b>FADATE</b>  Locale option FADATE suboption (Y/N). Refer to "Locale" on page 480 for information about this suboption.
115057	(1C171)	CHAR	R/O	5400	<b>IDISYSDB_PRE</b>  IDISYSDB preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDISYSDB DDname.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
120457	(1D689)	CHAR	R/W	5400	<b>IDISYSDB_JOB</b>  IDISYSDB data sets specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDISYSDB data sets specified via DataSets options in the user options file.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.
125857	(1EBA1)	CHAR	R/W	5400	<b>IDISYSDB_CFG</b>  IDISYSDB data sets specified in IDICNF00 config member. This field is initialized by Fault Analyzer to contain all IDISYSDB data sets specified via DataSets options in the IDICNF00 config member.  Refer to CTL.IDIADATA_PRE "Field format and usage" for more information.

## ENV - Common exit environment information

Table 16. ENV data area

Offsets Dec	Offsets Hex	Type	Access	Len	Name and description	See notes at end of table
0	(0)	CHAR	R/O	4	<b>VERSION</b>	
					Parameter list version (currently 0005).	
					Summary of changes:	
					0003    Support for 6-digit CICS system abend codes.	
					0004    Change of LOCK_FLAG from 1 to 2 characters to support fault entry expiration control.	
4	(4)	CHAR	R/O	1	<b>EXIT_CALL_TYPE</b>	
					User exit call type that indicates the type of exit being invoked as one of the following:	
					C       Analysis Control	
					L       Compiler Listing Read	
					R       Batch Report Tailoring	
					M       Message and Abend Code Explanation	
					F       Formatting	
					E       End Processing	
					N       Notification	
					I       IDIUTIL Import	
					D       IDIUTIL Delete	
					H       IDIUTIL ListHF	
					X       Dump registration Analysis Control exit	
					Y       Dump registration Notification exit	
					Z       Fault entry refresh End Processing exit	
					This information permits a user exit to be 'common' across exit types.	
5	(5)	CHAR	R/O	8	<b>FAULT_ID</b>	
					Fault ID.	
					For an End Processing user exit, and for a Notification user exit when a fault is a duplicate (NFY.NFYTYPE='N' or 'F'), this field contains the duplicate fault ID.	
					For a Notification user exit when a fault is not a duplicate (NFY.NFYTYPE='C' or 'R'), this field contains the assigned fault ID.	
					For all other exits, this field is not initialized.	
13	(D)	CHAR	R/O	10	<b>ABEND_DATE</b>	
					Date of abend in the format YYYY/MM/DD.	
23	(17)	CHAR	R/O	8	<b>ABEND_TIME</b>	
					Time of abend in the format HH:MM:SS (24-hour clock value).	

## ENV data area

Table 16. ENV data area (continued)

Offsets Dec	Type Hex	Access	Len	Name and description	See notes at end of table
31	(1F) CHAR	R/O	1	<b>REALTIME</b>  Real-time execution (Y/N).	
32	(20) CHAR	R/O	8	<b>SYSTEM_NAME</b>  System name: <ul style="list-style-type: none"><li>• When ENV.VERSION is 1, this field contains the APPLID for CICS transaction faults and the MVS system name for all other fault types.</li><li>• When ENV.VERSION is greater than 1, this field always contains the MVS system name while the CICS transaction application ID is provided in ENV.APPLID instead.</li></ul>	
40	(28) CHAR	R/O	8	<b>JOB_NAME</b>  Job/started task name.	
48	(30) CHAR	R/O	8	<b>EXEC_PGM_NAME</b>  EXEC program name.	2
56	(38) CHAR	R/O	8	<b>USER_ID</b>  User ID.	2
64	(40) CHAR	R/O	4	(Reserved)	
68	(44) CHAR	R/O	8	<b>ABEND_MODULE_NAME</b>  ABEND module name. This value identifies the module name where the initial (if more than one) abend occurred.	
76	(4C) CHAR	R/O	4	<b>CICS_TRANSACTION_ID</b>  CICS transaction ID.	
80	(50) CHAR	R/O	5	<b>CICS_TASK_NUMBER</b>  CICS task number.	
85	(55) CHAR	R/O	1	<b>JOB_TYPE</b>  The type of job being analyzed as one of the following: <b>B</b> Batch job or MVS dump analyzed interactively by way of the <b>File</b> menu option 5. <b>S</b> Started task <b>T</b> TSO <b>C</b> CICS transaction <b>I</b> CICS system dump analysis, including dump registration of CICS system dump <b>D</b> Dump registration (other than CICS system dump)	

Table 16. ENV data area (continued)

Offsets		Type	Access	Len	Name and description	See notes at end of table
Dec	Hex					
86	(56)	CHAR	R/O	1	<b>JOB_CLASS</b> Job execution class.	2
87	(57)	CHAR	R/O	3	<b>ACCOUNTING_FIELDS</b> Number of job accounting fields (nnn) (from JCT ACTJNFLD).	2
90	(5A)	CHAR	R/O	144	<b>ACCOUNTING_INFO</b> Job accounting information (from JCT ACTACCNT).  For job accounting, this field contains a one-byte length field, followed by the content of the field (repeated as many times as the number of fields shown in ENV.ACOUNTING_FIELDS).  For step accounting, this field contains these items (in order) for each step: <ul style="list-style-type: none"><li>• A three-byte maximum step running time.</li><li>• A one-byte number of fields in step.</li><li>• A one-byte length field.</li><li>• The content of the field.</li></ul> The last two fields repeat as many times as the number of fields in byte 4 indicates.  Any non-printable characters (such as the binary field length value) are shown as periods.	2
234	(EA)	CHAR	R/W	4	<b>USER_1</b>  User field 1. This field can be used to pass information from one user exit to another. Fault Analyzer does not reinitialize this field between calls to user exits and no upper case translation is performed. Truncation by null character (X'00') of this field is not permitted.	
238	(EE)	CHAR	R/W	4	<b>USER_2</b>  User field 2. Same as USER_1.	
242	(F2)	CHAR	R/O	1	(Reserved)	

## ENV data area

Table 16. ENV data area (continued)

Offsets	Type	Access	Len	Name and description	See notes at end of table
Dec	Hex				
243	(F3)	CHAR	R/W	1 <b>LOOPPROTECTION_OPT</b>  LoopProtection option (Y/N). <ul style="list-style-type: none"><li>• When set to Y, this value is equivalent to the LoopProtection option being in effect.</li><li>• When set to N, this value is equivalent to the NoLoopProtection option being in effect.</li></ul> It is only possible to deactivate the loop/wait protection feature of Fault Analyzer by setting this field to N. Setting this field to Y is ignored.  Refer to "LoopProtection" on page 481 for more information about this option.	
244	(F4)	CHAR	R/O	4 (Reserved)	
248	(F8)	ADDRESS	R/O	4 <b>WRITE_ROUTINE_EP</b>  Write routine entry-point address.	
252	(FC)	CHAR	R/O	4 (Reserved—always contains X'00000000')	
256	(100)	CHAR	R/O	1 <b>INVOCATION_EXIT</b>  The type of invocation exit used to invoke Fault Analyzer implicitly for real-time abend analysis, or explicitly using IDISNAP:  C CICS XPCABND exit (IDIXCX52 or IDIXCX53) D CICS XDUREQ exit (IDIXCX52 or IDIXCX53) E CICS LE CEECXTAN exit (IDIXCCEE) L LE CEEEXTAN exit (IDIXCEE) M MVS IEAVTABX change options/suppress dump exit (IDIXDCAP) S Fault Analyzer program SNAP interface (IDISNAP) P MVS IEAVTSEL post dump exit (IDIXTSEL) (Fault Analyzer dump registration)	
257	(101)	CHAR	R/O	8 <b>STEP_NAME</b>  Job/started task step name.	2
265	(109)	CHAR	R/O	8 <b>JOB_ID</b>  JES job ID.	2
273	(111)	CHAR	R/O	8 <b>IMS_PROGRAM_NAME</b>  IMS program name. This name is available if the environment has a DFSPRPX0 module loaded.	2

Table 16. ENV data area (continued)

Offsets	Type	Access	Len	Name and description	See notes at end of table
Dec	Hex				
281	(119)	CHAR	R/W	8 <b>USER_NAME</b>  User name field.	
289	(121)	CHAR	R/W	40 <b>USER_TITLE</b>  User title field.	
329	(149)	CHAR	R/O	8 <b>APPLID</b>  Application ID. Only applicable to CICS transaction faults (ENV.JOB_TYPE = C) and when ENV.VERSION is greater than 1, in which case it contains the associated CICS APPLID.	
337	(151)	CHAR	R/O	4 <b>TERMID</b>  CICS terminal ID.	
341	(155)	CHAR	R/O	8 <b>NETNAME</b>  CICS terminal netname.	
349	(15D)	CHAR	R/O	8 <b>TCB_ADDRESS</b>  Analyzed TCB address. <b>Note:</b> The TCB address for CICS transaction abends points to the QR TCB, which is no longer running the abending transaction.	
357	(165)	CHAR	R/O	8 <b>CSA_ADDRESS</b>  CICS CSA address.  This field is only available for CICS transaction abend or CICS system dump analysis.	
365	(16D)	CHAR	R/O	8 <b>TCA_ADDRESS</b>  CICS TCA address.  This field is only available for CICS transaction abend analysis.	

## ENV data area

Table 16. ENV data area (continued)

Offsets Dec	Offsets Hex	Type	Access	Len	Name and description	See notes at end of table
373	(175)	CHAR	R/W	44	<b>IDIHIST</b>  Fault history file name.  This field is not used for any IDIUTIL batch utility user exits. For all other user exit types, this field is initialized by Fault Analyzer to the history file name provided via the IDIHIST DDname (either preallocated or via the DataSets option).  During real-time processing, an Analysis Control or End Processing user exit might choose to change this data set name to that of another history file, which is subsequently used instead. For all other user exit types or processing modes, this field is read-only.	
417	(1A1)	CHAR	R/O	6	<b>ABEND_CODE</b>  For a Notification user exit invoked for a duplicate fault (NFY.NFYTYPE='N' or 'F'), this name is the name of the history file in which the duplicate fault identified by ENVFAULT_ID was found.	
423	(1A7)	CHAR	R/O	6	<b>CPU_HSECONDS</b>  Total CPU time used by Fault Analyzer in 1/100s of a second at the end of generating the analysis report.  <b>Note:</b> This field is available for use with IDIUTIL batch utility user exits only, and data is only provided for PDSE history files that are managed by the IDIS subsystem using the PARM='UPDINDEX' option (see "Caching of history file \$\$INDEX data" on page 240).	
429	(1AD)	CHAR	R/O	9	<b>CICS_VRM</b>  CICS release level in VnnRnnMnn format	3
438	(1B6)	CHAR	R/O	9	<b>DB2_VRM</b>  DB2 release level in VnnRnnMnn format	3
447	(1BF)	CHAR	R/O	9	<b>IMS_VRM</b>  IMS release level in VnnRnnMnn format	3

Table 16. ENV data area (continued)

Offsets		Type	Access	Len	Name and description	See notes at end of table
Dec	Hex					
456	(1C8)	CHAR	R/O	9	<b>ZOS_VRM</b> z/OS release level in VnnRnnMnn format	3
465	(1D1)	CHAR	R/W	2	<b>LOCK_FLAG</b>  Fault entry lock flag. The purpose of this flag is to provide a mechanism that prevents accidental deletion of the current fault entry. For more information about this flag, including specification of fault entry expiration control, see "Viewing fault entry information" on page 82.  By default, the lock flag is set to a blank, which does not prevent deletion of the fault entry.  This flag can be modified by any user exit. However, changes are only reflected in the history file fault entry when the user exit is invoked during real-time analysis processing, or when creating a new fault entry from interactive reanalysis of an MVS dump data set.	
467	(1D3)	CHAR	R/O	5	<b>DUPLICATE_COUNT</b>  Any printable character can be specified for this field: <ul style="list-style-type: none"><li>• If a non-printable character is specified, then it is changed to a '/'.</li><li>• If a lowercase character is specified, then it is translated to uppercase.</li></ul>	
472	(1D8)	CHAR	R/O	8	<b>POF_MODULE_NAME</b>  Total number of duplicates, not including the current fault.	
480	(1E0)	CHAR	R/O	10	<b>POF_MODULE_LKED_DATE</b>  Point-of-failure module link-edit date in the format YYYY/MM/DD.	

## ENV data area

Table 16. ENV data area (continued)

Offsets Dec	Offsets Hex	Type	Access	Len	Name and description	See notes at end of table
490	(1EA)	CHAR	R/O	8	<b>POF_MODULE_LKED_TIME</b>  Point-of-failure module link-edit time in the format HH:MM:SS.	
498	(1F2)	CHAR	R/O	8	<b>POF_CSECT_NAME</b>  Point-of-failure CSECT name.	
506	(1FA)	CHAR	R/O	10	<b>POF_CSECT_OFFSET</b>  Point-of-failure entry-point, program/CSECT or load module offset (decimal).	
516	(204)	CHAR	R/O	44	<b>POF_LOADED_FROM</b>  Data set name from where the point-of-failure program was loaded.	
560	(230)	CHAR	R/O	44	<b>EXEC_LOADED_FROM</b>  Data set name from where the EXEC program was loaded.	
604	(25C)	CHAR	R/O	10	<b>DUP_DATE</b>  Date of most recent duplicate fault in the format YYYY/MM/DD.	
614	(266)	CHAR	R/O	8	<b>DUP_TIME</b>  Time of most recent duplicate fault in the format HH:MM:SS (24-hour clock value).	
622	(26E)	CHAR	R/O	8	<b>GROUP_ID</b>  Security server default group ID.	2
630	(276)	CHAR	R/O	6	<b>INVOCATION_ABEND_CODE</b>  The final (or only) abend code, which is the abend code for which Fault Analyzer was invoked: <ul style="list-style-type: none"><li>• If ENV_JOB_TYPE = C, then this code is a 4-character CICS transaction abend code.</li><li>• Else if ENV_JOB_TYPE = I, then this code is a 6-character CICS system abend code.</li><li>• Otherwise, it is either a system abend code (Sxxx) or a user abend code (Unnnn).</li></ul>	
636	(27C)	CHAR	R/O	10	<b>MINIDUMP_PAGES</b>  Number of minidump pages (nnnnnnnnnn). <b>Note:</b> For real-time processing, more minidump pages resulting from storage that is referenced during execution of a Formatting user exit are not included in this value.	

Table 16. ENV data area (continued)

Offsets		Type	Access	Len	Name and description	See notes at end of table
Dec	Hex					
646	(286)	CHAR	R/O	8	IDIRLOAD_DD	1
					IDIRLOAD DDname.	
					By default, this field is initialized to IDIRLOAD, but can be changed by an Analysis Control user exit to another DDname to which a load library has been allocated.	
					The specified DDname is not case sensitive.	
654	(28E)	CHAR	R/O	8	ABEND_REASON_CODE	
					Hexadecimal reason code that is associated with ENV.ABEND_CODE.	
					Available to all user exits that run following "Analysis" processing as per Figure 144 on page 360.	
662	(296)	CHAR	R/O	8	LOCK_USERID	
					The user ID who last changed the ENV.LOCK_FLAG field.	
670	(29E)	CHAR	R/O	10	ORIGINAL_DATE	4
					Date of original fault in the format YYYY/MM/DD.	
680	(2A8)	CHAR	R/O	8	ORIGINAL_TIME	4
					Time of original fault in the format HH:MM:SS (24-hour clock value).	
688	(280)	CHAR	R/O	852	(Reserved)	

## Notes:

- 1 Not applicable to real-time processing.
- 2 Not available to dump registration user exits.
- 3 Not available at the time of calling the Analysis Control user exit.
- 4 Only available if the fault is a duplicate.

**EPC - End Processing user exit parameter list**

Table 17. EPC data area

Offsets		Type	Access	Len	Name and description
Dec	Hex				
0	(0)	CHAR	R/O	4	VERSION
					Parameter list version (currently 0002).
4	(4)	CHAR	R/O	4	(Reserved)

## EPC data area

Table 17. EPC data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
8	(8) CHAR	R/W	1	<b>IS_DUPLICATE</b> This parameter is a duplicate fault (Y/N).
9	(9) CHAR	R/W	1	<b>SUPPRESS_MINIDUMP</b> Suppress minidump (Y/N). This flag is set by Fault Analyzer based on the MaxMinidumpPages option setting and the expected minidump pages for this fault. It can be overridden by an End Processing user exit.
10	(A) CHAR	R/O	1	(Reserved)
11	(B) CHAR	R/W	1	<b>SUPPRESS_FAULT_ENTRY</b> If real-time, suppress history file fault entry, or if fault entry refresh, do not update the history file fault entry (Y/N).
12	(C) CHAR	R/O	15	(Reserved)
26	(1A) CHAR	R/O	5	<b>MINUTES_SINCE_LAST_DUP</b> Number of minutes elapsed since recording of last duplicate fault (nnnnn). If blank, no duplicate fault was found. <b>Note:</b> The maximum value of 99999 is used whenever the number of minutes exceeds the limit of this field.
31	(1F) CHAR	R/O	1	<b>ANALYSIS_SUCCESSFUL</b> Successful analysis (Y/N). The criteria for successful analysis are the following: <ul style="list-style-type: none"><li>• No error messages issued.</li><li>• Identification of the source line of code for the point of failure.</li></ul>
32	(20) CHAR	R/O	88	(Reserved)
120	(78) CHAR	R/W	1	<b>SUPPRESS_DUMP</b> Suppress dump (Y/N). This flag affects the suppression of the MVS system dump or CICS transaction dump. For details, see "Dump suppression" on page 15.
121	(79) CHAR	R/O	63	(Reserved)

## LST - Compiler Listing Read user exit parameter list

Table 18. LST data area

Offsets	Type	Access	Len	Name and description
Dec	Hex			
0	(0) CHAR	R/O	4	<b>VERSION</b> Parameter list version (currently 0001).

Table 18. LST data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
4	(4)	CHAR	R/O	8 <b>MODULE_NAME</b> Module name.  This name is the name of the load module containing the CSECT identified in LST.CSECT_NAME.
12	(C)	CHAR	R/O	8 <b>CSECT_NAME</b> CSECT name.  This name is the name of the CSECT containing the program identified in LST.PROGRAM_NAME.
20	(14)	CHAR	R/O	256 <b>EP_NAME</b> Entry point name (truncated to 256 chars).
276	(114)	CHAR	R/O	10 <b>COMPILE_DATE</b> Compile date in the format YYYY/MM/DD.
286	(11E)	CHAR	R/O	8 <b>COMPILE_TIME</b> Compile time in the format HH:MM:SS.
294	(126)	CHAR	R/O	1 <b>LISTING_TYPE</b> Compiler listing or assembler SYSADATA file (L), or side file (S).
295	(127)	CHAR	R/O	12 <b>LANGUAGE_TYPE</b> Language type: • Assembler • C/C++ • COBOL • OS/VS COBOL • PL/I • Entprs PL/I
307	(133)	CHAR	R/O	4 <b>RECFM</b> Record format.
311	(137)	CHAR	R/O	5 <b>LRECL</b> Logical record length (nnnnn).
316	(13C)	CHAR	R/W	5 <b>DATA_LENGTH</b> Data length of variable length record (nnnnn). This field specifies the length of the record placed in DATA_BUFFER.
321	(141)	CHAR	R/W	1 <b>DISREGARD_EXIT_LISTING</b> Ignore compiler listing or side-file supplied by the exit (Y/N). This field is always initialized to 'N' by Fault Analyzer. If 'Y' is returned, Fault Analyzer disregards any data that might have been provided and continues the search for the listing or side-file through the normal search path.

## LST data area

Table 18. LST data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
322	(142)	CHAR	R/O	8	<b>PROGRAM_NAME</b> Program name.
330	(14A)	CHAR	R/O	10	<b>PROGRAM_LENGTH</b> Program length in bytes (decimal).
340	(154)	CHAR	R/W	1	<b>DATA_BUFFER_DSN</b> Data buffer contains data set name (Y/N)
341	(155)	CHAR	R/O	44	<b>LOAD_MODULE_DSN</b> Load module data set name.  This name is the name of the data set from which the load module identified in LST MODULE_NAME was loaded.
385	(181)	CHAR	R/O	5	(Reserved)
390	(186)	CHAR	R/W	8188	<b>DATA_BUFFER</b> Data buffer.  No upper case translation is performed on the contents of this field. Truncation by null character (X'00') of this field is not permitted. For variable-length records, the length must be provided in the DATA_LENGTH field. For fixed-length records, the length is expected to match the LRECL.  If DATA_BUFFER_DSN is set to Y, then it is expected that this field contains the name of a data set (with member name following in parenthesis if partitioned) that contains the compiler listing or side file as appropriate for LISTING_TYPE. Refer to "Compiler Listing Read user exit" on page 372 for extra data set name requirements.

## NFY - Notification user exit parameter list

Table 19. NFY data area

Offsets		Type	Access	Len	Name and description
Dec	Hex				
0	(0)	CHAR	R/O	4	<b>VERSION</b> Parameter list version (currently 0002).
4	(4)	CHAR	R/O	45	(Reserved)
49	(31)	CHAR	R/O	1024	<b>SYNOPSIS</b> Fault analysis synopsis. Individual lines of the synopsis are delimited by new-line characters (X'15').  Fault Analyzer permits a buffered data format that is used if the size of the synopsis exceeds the maximum that can be contained in this field. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.

Table 19. NFY data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
1073	(431)	CHAR	R/O	1	NFYTYPE
				C	Fault created
				R	Recovery fault recording
				N	NoDup(Normal) duplicate
				F	NoDup(CICSfast) or NoDup(ImageFast) duplicate
1074	(432)	CHAR	R/O	8	DUPCOUNT
					The number of new duplicates during this 30-second recording period when NFYTYPE is set to 'F'. Always 1 when NFYTYPE is set to 'N'. Not applicable for other values of NFYTYPE.
1082	(43A)	CHAR	R/O	55	(Reserved)

## UFM - Formatting user exit parameter list

Table 20. UFM data area

Offsets		Type	Access	Len	Name and description
Dec	Hex				
0	(0)	CHAR	R/O	4	VERSION
					Parameter list version (currently 0001).
4	(4)	CHAR	R/W	100	USEROPTIONTITLE
					Report section heading for output from all Formatting user exits run using the Exits option. Initialized to the heading set by any previously called Formatting user exit. The initial default is "User".
104	(68)	CHAR	R/O	91	(Reserved)
195	(C3)	CHAR	R/O	5	NUM_EVENTS
					Total number of events (decimal).
All fields from here on are populated with data for a single event only. To populate with data for another event, use the IDIEventInfo command.					
200	(C8)	CHAR	R/W	5	EVENT_NO
					Current event number (nnnnn).
205	(CD)	CHAR	R/O	5	NEXT_EVENT_NO
					Next available event number (decimal).
210	(D2)	CHAR	R/O	5	PREVIOUS_EVENT_NO
					Previous available event number (decimal).
215	(D7)	CHAR	R/O	1	POF
					Point of failure (Y/N).

## UFM data area

Table 20. UFM data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
216	(D8)	CHAR	R/O	30 <b>EVENT_TYPE</b>  Event type in the same format as shown in the Event Summary section of the analysis report, for example, "Abend S0C7".  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
				<b>Japanese feature note</b>
				If Language(JPN) is in effect, then the event type description provided in this field is subject to translation into Japanese.
				<b>End of Japanese feature note</b>
				<b>Korean feature note</b>
				If Language(KOR) is in effect, then the event type description provided in this field is subject to translation into Korean.
				<b>End of Korean feature note</b>
246	(F6)	CHAR	R/O	12 <b>MODULE_NAME</b>  Module name.  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
258	(102)	CHAR	R/O	8 <b>MODULE_ADDRESS</b>  Module address.
266	(10A)	CHAR	R/O	8 <b>MODULE_LENGTH</b>  Module length (hexadecimal).
274	(112)	CHAR	R/O	12 <b>PROGRAM_NAME</b>  Program name.  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
286	(11E)	CHAR	R/O	8 <b>PROGRAM_ADDRESS</b>  Program address.
294	(126)	CHAR	R/O	8 <b>PROGRAM_LENGTH</b>  Program length (hexadecimal).

Table 20. UFM data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
302	(12E)	CHAR	R/O	12	<b>EP_NAME</b> Entry point name.  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
314	(13A)	CHAR	R/O	8	<b>EP_ADDRESS</b> Entry point address.
322	(142)	CHAR	R/O	64	<b>EVENT_LOCATION</b> Event location in the same format as shown in the Event Summary section of the analysis report, for example, "L#31 P+3D4".  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
386	(182)	CHAR	R/O	44	<b>LOADED_FROM</b> Information about from where the module was loaded in the same format as shown in the Event Summary section of the analysis report, for example, a data set name.  If data for this field exceeds the field size, then a buffered data format is used. For details, see "Non-REXX user exit buffered data format" on page 505. The format of this field is transparent to users of REXX exits.
430	(1AE)	CHAR	R/O	8	<b>INSTRUCTION_ADDRESS</b> The event instruction address.
438	(1B6)	CHAR	R/O	2	<b>AMODE</b> The event addressing mode (24/31).
440	(1B8)	CHAR	R/O	16	<b>PSW</b> The event PSW.
456	(1C8)	CHAR	R/O	8	<b>GPREG0</b> General purpose register 0.
464	(1D0)	CHAR	R/O	8	<b>GPREG1</b> General purpose register 1.
472	(1D8)	CHAR	R/O	8	<b>GPREG2</b> General purpose register 2.
480	(1E0)	CHAR	R/O	8	<b>GPREG3</b> General purpose register 3.

## UFM data area

Table 20. UFM data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
488	(1E8)	CHAR	R/O	8 <b>GPREG4</b> General purpose register 4.
496	(1F0)	CHAR	R/O	8 <b>GPREG5</b> General purpose register 5.
504	(1F8)	CHAR	R/O	8 <b>GPREG6</b> General purpose register 6.
512	(200)	CHAR	R/O	8 <b>GPREG7</b> General purpose register 7.
520	(208)	CHAR	R/O	8 <b>GPREG8</b> General purpose register 8.
528	(210)	CHAR	R/O	8 <b>GPREG9</b> General purpose register 9.
536	(218)	CHAR	R/O	8 <b>GPREG10</b> General purpose register 10.
544	(220)	CHAR	R/O	8 <b>GPREG11</b> General purpose register 11.
552	(228)	CHAR	R/O	8 <b>GPREG12</b> General purpose register 12.
560	(230)	CHAR	R/O	8 <b>GPREG13</b> General purpose register 13.
568	(238)	CHAR	R/O	8 <b>GPREG14</b> General purpose register 14.
576	(240)	CHAR	R/O	8 <b>GPREG15</b> General purpose register 15.
584	(248)	CHAR	R/O	8 <b>AREG_DATA_ADDRESS</b> Address of storage area containing access registers in hexadecimal format (AR0 through AR15).
592	(250)	CHAR	R/O	138 (Reserved)
730	(2DA)	CHAR	R/W	5 <b>DATA_LENGTH</b> Data length (nnnnn). This field specifies the length of the record placed in UFM.DATA_BUFFER.

Table 20. UFM data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
735	(2DF)	CHAR	R/W	1024 <b>DATA_BUFFER</b>  Data buffer.  No upper case translation is performed on the contents of this field. Truncation by null character (X'00') of this field is not permitted. The length must be provided in the UFM.DATA_LENGTH field.  <b>Note:</b> The purpose of this field is to serve as a record buffer when passing records back to Fault Analyzer from a load module user exit using the ENV.WRITE_ROUTINE_EP program. For details on how to use this buffer, see "Formatting user exit" on page 381. REXX user exits need not use this field as data can be passed back to Fault Analyzer directly using the IDIWRITE command.
1759	(6DF)	CHAR	R/O	1 (Reserved)
1760	(6E0)	CHAR	R/O	16 <b>FPREG0</b>  Floating-point register 0.
1776	(6F0)	CHAR	R/O	16 <b>FPREG1</b>  Floating-point register 1.
1792	(700)	CHAR	R/O	16 <b>FPREG2</b>  Floating-point register 2.
1808	(710)	CHAR	R/O	16 <b>FPREG3</b>  Floating-point register 3.
1824	(720)	CHAR	R/O	16 <b>FPREG4</b>  Floating-point register 4.
1840	(730)	CHAR	R/O	16 <b>FPREG5</b>  Floating-point register 5.
1856	(740)	CHAR	R/O	16 <b>FPREG6</b>  Floating-point register 6.
1872	(750)	CHAR	R/O	16 <b>FPREG7</b>  Floating-point register 7.
1888	(760)	CHAR	R/O	16 <b>FPREG8</b>  Floating-point register 8.
1904	(770)	CHAR	R/O	16 <b>FPREG9</b>  Floating-point register 9.
1920	(780)	CHAR	R/O	16 <b>FPREG10</b>  Floating-point register 10.
1936	(790)	CHAR	R/O	16 <b>FPREG11</b>  Floating-point register 11.

## UFM data area

Table 20. UFM data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
1952	(7A0)	CHAR	R/O	16 <b>FPREG12</b> Floating-point register 12.
1968	(7B0)	CHAR	R/O	16 <b>FPREG13</b> Floating-point register 13.
1984	(7C0)	CHAR	R/O	16 <b>FPREG14</b> Floating-point register 14.
2000	(7D0)	CHAR	R/O	16 <b>FPREG15</b> Floating-point register 15.
2016	(7E0)	CHAR	R/O	8 <b>FPCR</b> Floating-point control register.

## UTL - IDIUTIL Batch Utility user exit parameter list

Table 21. UTL data area

Offsets	Type	Access	Len	Name and description
Dec	Hex			
0	(0)	CHAR	R/O	4 <b>VERSION</b> Parameter list version (currently 0002).
4	(4)	CHAR	R/O	44 (Reserved)
48	(30)	CHAR	R/W	1 <b>PERFORM_ACTION</b> Perform the utility action on this fault entry (Y/N).
49	(31)	CHAR	R/O	64 (Reserved)

## XPL - Message and Abend Code Explanation user exit parameter list

Table 22. XPL data area

Offsets	Type	Access	Len	Name and description
Dec	Hex			
0	(0)	CHAR	R/O	4 <b>VERSION</b> Parameter list version (currently 0001).
4	(4)	CHAR	R/O	125 <b>MESSAGE_TEXT1</b> Single-line WTO or first multi-line WTO message text.
129	(81)	CHAR	R/O	70 <b>MESSAGE_TEXT2</b> Multi-line WTO message text line 2.
199	(C7)	CHAR	R/O	70 <b>MESSAGE_TEXT3</b> Multi-line WTO message text line 3.

Table 22. XPL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
269	(10D)	CHAR	R/O	70	<b>MESSAGE_TEXT4</b> Multi-line WTO message text line 4.
339	(153)	CHAR	R/O	70	<b>MESSAGE_TEXT5</b> Multi-line WTO message text line 5.
409	(199)	CHAR	R/O	70	<b>MESSAGE_TEXT6</b> Multi-line WTO message text line 6.
479	(1DF)	CHAR	R/O	70	<b>MESSAGE_TEXT7</b> Multi-line WTO message text line 7.
549	(225)	CHAR	R/O	70	<b>MESSAGE_TEXT8</b> Multi-line WTO message text line 8.
619	(26B)	CHAR	R/O	70	<b>MESSAGE_TEXT9</b> Multi-line WTO message text line 9.
689	(2B1)	CHAR	R/O	70	<b>MESSAGE_TEXT10</b> Multi-line WTO message text line 10.
759	(2F7)	CHAR	R/O	4	(Reserved)
763	(2FB)	CHAR	R/O	8	<b>ABEND_REASON_CODE</b> ABEND reason code.
771	(303)	CHAR	R/O	8	<b>ABEND_MODULE_NAME</b> ABEND module name.
779	(30B)	CHAR	R/O	1	<b>ABEND_TYPE</b> ABEND type: C CICS transaction abend D CICS dump code S System U User
780	(30C)	CHAR	R/W	5	<b>DATA_LENGTH</b> Data length (nnnnn). This field specifies the length of the record placed in DATA_BUFFER.

## XPL data area

Table 22. XPL data area (continued)

Offsets	Type	Access	Len	Name and description
Dec	Hex			
785	(311)	CHAR	R/W	256 <b>DATA_BUFFER</b>  Data buffer.  No upper case translation is performed on the contents of this field. Truncation by null character (X'00') of this field is not permitted. The length must be provided in the DATA_LENGTH field unless a REXX EXEC variable name is used on the IDIWRITE command. <b>Note:</b> The purpose of this field is to serve as a record buffer when passing records back to Fault Analyzer from the user exit using IDIWRITE (REXX) or the ENV.WRITE_ROUTINE_EP program (non-REXX). For details on this, see "Message and Abend Code Explanation user exit" on page 377.
1041	(411)	CHAR	R/O	1 <b>EXPLANATION_AVAILABLE</b>  Flag to indicate whether Fault Analyzer was able to provide the explanation of the message or abend code (Y/N).
1042	(412)	CHAR	R/O	6 <b>ABEND_CODE</b>  ABEND code.
1048	(418)	CHAR	R/O	58 (Reserved)

---

## Chapter 35. Return codes

This chapter describes the return codes issued by Fault Analyzer.

---

### Batch reanalysis (IDIDA)

The following return codes might be received when performing batch fault reanalysis:

RC	Meaning
----	---------

0	<ul style="list-style-type: none"><li>One or more informational messages might have been issued (message numbers that are suffixed by 'T').</li></ul>
---	---

2 or 4	<ul style="list-style-type: none"><li>One or more warning messages has been issued (message numbers that are suffixed by 'W').</li><li>Informational messages might also have been issued.</li></ul>
--------	--

8	<ul style="list-style-type: none"><li>One or more error messages has been issued (message numbers that are suffixed by 'E').</li><li>Informational and warning messages might also have been issued.</li></ul>
---	--

12	<ul style="list-style-type: none"><li>One or more severe messages has been issued (message numbers that are suffixed by 'S').</li><li>Informational, warning and error messages might also have been issued.</li></ul>
----	--

---

### IDIUTIL batch utility

The following return codes are issued by the IDIUTIL batch utility:

RC	Meaning
----	---------

0	Successful completion.
---	------------------------

4	One or more errors occurred, each identified by a message written to the SYSPRINT DDname.
---	---

---

### IDILANGX

For return codes issued by IDILANGX, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.



---

# Chapter 36. Messages

This chapter describes the messages that are issued by Fault Analyzer and IDILANGX.

---

## Fault Analyzer messages

These messages are issued by Fault Analyzer.

---

**IDI0001I**    *Fault Analyzer version-info invoked by exit-name using config-member*

**Explanation:** Fault Analyzer was invoked for real-time analysis. In the message text:

- *version-info* provides information about the version of Fault Analyzer used and the current maintenance level.
- *exit-name* identifies the exit which invoked Fault Analyzer.
- *config-member* is the data set and member name containing the parmlib configuration options, or indicates the use of default options if no configuration member could be found.

**System action:** Normal processing continues.

**User response:** None

---

**IDI0002I**    *analysis-summary*

**Explanation:** Fault analysis has completed. In the message text, *analysis-summary* provides a brief summary of the problem.

The analysis summary is typically presented in the format:

  [*point-of-failure*:] *symptom*

where:

*point-of-failure*

The point in the user application at which the error occurred, or where control left the user application prior to the error. If available, module name, program or CSECT name, offset, and source line number or compiler listing statement number is provided.

*symptom*

The type of error that occurred (for example, the initial abend code).

**System action:** Normal processing continues.

**User response:** None

---

**IDI0003I**    *Fault ID faultid assigned in history file history-file-name*

**Explanation:** Fault Analyzer has completed real-time analysis and has assigned the fault identifier *faultid* in the history file *history-file-name* to this abend.

**System action:** Processing has ended.

**User response:** None

---

**IDI0004S**    *The input dump data set data-set-name could not be opened because: reason*

**Explanation:** The dump data set identified by *data-set-name* could not be opened for reanalysis for the reason that is identified in *reason*.

**System action:** Processing terminates.

**User response:** Correct the data set name and resubmit the job.

---

**IDI0005S**    *module-name:line-number Storage allocation for dec-count (X'hex-count') bytes failed - processing terminated*

**Explanation:** An out-of-storage condition has occurred.

**System action:** Processing terminates.

**User response:** Specify a larger region size and resubmit the job—see “Storage recommendations” on page 222 for more information. If using TSO, specify a larger region size on the logon panel.

Fault Analyzer deliberately attempts not to use up all available 31-bit storage before issuing this message, as using up all storage might cause MVS to use 24-bit storage instead with possibly disastrous consequences for non-terminating address spaces, such as CICS regions.

---

**IDI0006E**    *Open of context data set data-set-name failed because: reason*

**Explanation:** The data set identified by *data-set-name* could not be opened for the reason that is identified in *reason*.

The context in which the data set was attempted opened is provided in *context*. This context might be an

## IDI0007S • IDI0015W

associated DDname, or some other description of the type of data set involved.

When *reason* is in the form of

DYNALLOC error=*error-code* info=*info-code*

then refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the error and info codes.

The reasons for some of the more common DYNALLOC info codes are as follows:

### Info code

#### Reason

- |      |                                       |
|------|---------------------------------------|
| 210  | Data set is allocated to another job. |
| 1708 | Data set not found.                   |

**System action:** Processing attempts to continue without the use of this data set.

**User response:** Determine the reason for the open failure and resubmit the job.

---

**IDI0007S    GETMAIN failed *resource-name***  
rc=*return-code*

**Explanation:** An out-of-storage condition has occurred.

**System action:** Processing terminates.

**User response:** Specify a larger region size and resubmit the job.

---

**IDI0008E    CSVINFO error, rc=*return-code***

**Explanation:** An error occurred using the CSVINFO service.

**System action:** Processing terminates.

**User response:** This error might be an internal error. Contact your IBM service representative.

---

**IDI0009E    IEWBUFF error**

**Explanation:** An error occurred while using the Binder.

**System action:** Processing terminates.

**User response:** This error might be an internal error. Contact your IBM service representative.

---

**IDI0010E    IEWBIND error *function* *module-name***  
rc=*reason-code*

**Explanation:** A call to the MVS Binder program was unsuccessful. In the message text, *function* identifies the type of function requested, *module-name* is the module attempted bound, and *reason-code* is the reason code returned by the Binder.

**Note:** The value in *reason-code* is the Binder API reason code. The Binder API reason codes can be found in *z/OS: MVS Program Management: Advanced Facilities*.

**System action:** Processing continues but analysis might be incomplete.

**User response:** This error might be an internal error. Contact your IBM service representative.

---

**IDI0011S    Abend *abend-code* occurred in Fault Analyzer analysis**

**Explanation:** An abend occurred in Fault Analyzer.

**System action:** Processing terminates.

**User response:** This error is an internal error. Contact your IBM service representative.

---

**IDI0012S    Abend *abend-code* occurred in abend exit processing,**

**Explanation:** An abend occurred in Fault Analyzer. This message is followed by message IDI0013S.

**System action:** Processing terminates.

**User response:** This error is an internal error. Contact your IBM service representative. Ensure that the SVC dump written by Fault Analyzer is kept for later analysis by IBM.

---

**IDI0013S    R15=r15-value PSW=program-status-word**  
**DCAPSUB=base-reg-value**

**Explanation:** This message follows message IDI0012S.

**System action:** See message IDI0012S.

**User response:** See message IDI0012S.

---

**IDI0014E    MTRACE error calling IEEMB879 for buffer accumulation, rc=rc.**

**Explanation:** An error occurred during MTRACE processing.

**System action:** Processing of MTRACE is terminated and information about console messages might be missing in the Fault History file.

**User response:** This error might be an internal error. Contact your IBM service representative.

---

**IDI0015W    Message *message-number* was not found for display**

**Explanation:** An unsuccessful attempt was made to issue the message identified by *message-number*.

**System action:** Processing continues.

**User response:** This error is an internal error. Contact your IBM service representative.

---

**IDI0016E Abend abend-code during fault history file OPEN/READ processing**

**Explanation:** During real-time processing, an abend occurred while attempting to OPEN or READ the history file.

**System action:** Processing continues, however, no fault entry is created and no duplicate counts is updated.

**User response:** Determine the reason for the abend.

---

**IDI0018W Parmlib member read error: reason**

**Explanation:** A problem occurred while attempting to open or read the IDICNF00 parmlib configuration member. The reason for the error is provided in *reason*.

**System action:** Processing continues without the use of the IDICNF00 parmlib member.

**User response:** Ensure that an IDICNF00 member exists in the logical parmlib concatenation, or in the alternative parmlib data set specified via the IDISCNF USERMOD.

---

**IDI0019W options-source syntax error on line line-number column column-number: reason**

**Explanation:** A syntax error was encountered while processing options that are specified in *options-source*, where *options-source* is one of the following:

**Analysis Control user exit**

The option was provided by an Analysis Control user exit.

**Environment variable \_IDI\_OPTS**

The option was provided through the \_IDI\_OPTS environment variable.

**Options line for interactive reanalysis**

The option was specified in the "Options line for interactive reanalysis" field on the Interactive Reanalysis Options display, which is shown when selecting "Interactive Reanalysis Options..." from the action bar Options pull-down menu.

**PARM field**

The option was specified in one of these places:

- The PARM field of the EXEC card for PGM=IDIDA in a batch reanalysis job.
- The **Options line for batch reanalysis** field on the Batch Reanalysis Options display. This field is shown when selecting **Batch Reanalysis Options...** from the action bar Options pull-down menu.

**Parmlib config member**

The option was specified in the IDICNFxx

parmlib member, or in a data set and member identified by an IDICNFUM user-options module.

**User options file**

The option was specified through the IDIOPTS DDname.

**System action:** Processing continues.

**User response:** Correct the error and resubmit your job.

---

**IDI0020W options-source contained invalid option option**

**Explanation:** An invalid option was encountered while processing options that are specified in *options-source*. For the possible values of *options-source*, see message IDI0019W.

**System action:** The option *option* is ignored and processing continues.

**User response:** Correct the error and resubmit your job.

---

**IDI0021W Allocation failed: DD=ddname  
DSN=data-set-name RC=return-code  
Error=error-code Info=info-code**

**Explanation:** Dynamic data set allocation failed.

**System action:** Processing continues without the data set identified in *data-set-name*.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

**IDI0022W Suboption(s) missing for options-source option option**

**Explanation:** An option with missing required suboptions(s) was encountered while processing options that are specified in *options-source*. For the possible values of *options-source*, see message IDI0019W.

**System action:** Processing continues with default value for option *option*.

**User response:** Correct the error and resubmit your job.

---

**IDI0023W Suboption(s) ignored for options-source option option**

**Explanation:** An option in *options-source* invalidly specified one or more suboptions when no suboptions were allowed. For the possible values of *options-source*, see message IDI0019W.

**System action:** Processing continues with default value for option *option*.

## IDI0024W • IDI0031W

**User response:** Correct the error and resubmit your job.

---

**IDI0024W Concatenation failed for**  
**DSN=***data-set-name* **to DD=***ddname*  
**RC=***return-code* **Error=***error-code* **Info=***info-code*

**Explanation:** Dynamic data set concatenation failed.

**System action:** Processing continues without the data set identified in *data-set-name*.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

**IDI0025W Suboption %s invalid for options-source option** *option*

**Explanation:** An option in *options-source* specified one or more invalid suboptions. For the possible values of *options-source*, see message IDI0019W.

**System action:** Processing continues with default value for option *option*.

**User response:** Correct the error and resubmit your job.

---

**IDI0026W Information retrieval failed for**  
**DD=***ddname* **RC=***return-code*  
**Error=***error-code* **Info=***info-code*

**Explanation:** Dynamic allocation information retrieval failed.

**System action:** Processing continues without the data set identified in *data-set-name*.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

**IDI0027E Fetch/load failed for module**  
*module-name*

**Explanation:** An attempt to fetch or load module *module-name* failed.

If *module-name* is prefixed by IPV, then the reason for the fetch/load failure is that the IBM Problem Determination Tools for z/OS Common Component Common Server was unable to load the specified module. These modules are provided by the Common Server and must be accessible via LINKLIST.

**System action:** If *module-name* is not prefixed by IPV, then processing continues without module *module-name*. However, depending on the functionality of the subject load module, processing might be incomplete.

If *module-name* is prefixed by IPV, then the Fault Analyzer process of the IBM Problem Determination Tools for z/OS Common Component Common Server

terminates, resulting in the inability to use the Fault Analyzer plug-in for Eclipse feature.

**User response:** Contact your systems programmer.

---

**IDI0028W Error reading user options file**

**Explanation:** An error occurred while reading the user options file via DDname IDIOPTS.

**System action:** Processing continues without the user options file.

**User response:** This error might be an internal error. Contact your IBM service representative.

---

**IDI0029W options-source options syntax error at**  
**offset** *offset*: *reason*

**Explanation:** A syntax error was encountered while processing options that are specified in *options-source*, where *options-source* is one of the following:

**PARM field**

This field is the JCL EXEC statement PARM field specified in a job that executes program IDIDA.

**Environment variable \_IDI\_OPTS**

This environment variable is set by the abending application.

**System action:** Processing continues.

**User response:** Correct the error and resubmit your job.

---

**IDI0030W Allocation to SYSOUT=(class,*n*,form-name) failed: DD=***ddname* **RC=***return-code*  
**Error=***error-code* **Info=***info-code*

**Explanation:** Dynamic allocation of JES spool data set failed.

**System action:** Processing continues without the DDname identified in *ddname*.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

**IDI0031W Open of DD=***ddname* **failed:** *reason*

**Explanation:** An unsuccessful attempt was made to open DDname *ddname*.

**System action:** Processing continues without the DDname identified in *ddname*.

**User response:** Refer to *reason* for a possible explanation.

---

**IDI0032W I/O error writing report: *reason***

**Explanation:** An I/O error occurred while writing the analysis report.

**System action:** Processing continues, however, the analysis report might be missing information.

**User response:** Refer to *reason* for a possible explanation.

---

**IDI0033E I/O error writing to *history-file-dsn*:  
System abend *abend-code*-*reason-code*  
*reason-text***

**Explanation:** An I/O error occurred while writing a fault history record to the history file data set shown in *history-file-dsn*.

**System action:** Processing continues, however, the history file does not contain information for the current job.

**User response:** Refer to *abend-code*, *reason-code* and *reason-text* for a possible explanation.

---

**IDI0034I Fault analysis skipped due to: *reason***

**Explanation:** A job was excluded from fault analysis for the reason that is identified in *reason* as one of the following:

- **EXCLUDE option specification (FAST)**

A match for the current fault was found during fast Exclude options processing (see “Fast Exclude options processing” on page 282).

- **Parmlib config member Exclude option specification**

An Exclude option that is specified in the IDICNF00 configuration member matched the current fault attributes. No subsequent Include or Exclude option specifications in a user options file matched, and no attempt was made to override the exclusion by an Analysis Control user exit.

- **User options file Exclude option specification**

An Exclude option that is specified in the IDIOPTS user options file matched the current fault attributes. No attempt was made to override the exclusion by an Analysis Control user exit.

- **Analysis Control user exit request**

No Exclude option specification in either the IDICNF00 configuration member or the IDIOPTS user options file matched the current fault attributes. Instead, an Analysis Control user exit requested the exclusion.

- **Abend *abend-code* during options processing**

An abend occurred during real-time options processing.

**System action:** No fault analysis is performed.

**User response:** If an abend occurred, and the problem

persists, contact your IBM service representative.

---

**IDI0035W No dump records found in *data-set-name***

**Explanation:** An attempt was made to read a dump data set during reanalysis, but the dump data set did not contain any data.

**System action:** Fault Analyzer attempts to continue without using the dump data set.

**User response:** Ensure that the correct dump data set was specified to Fault Analyzer.

---

**IDI0036E Abend *abend-code* during processing of  
*exit-type* user exit *exit-name*--refer to  
message explanation for problem  
determination information**

**Explanation:** An abend occurred during execution of a user exit. In the message text, *abend-code* is the type of abend that occurred, *exit-type* is the type of user exit that abended, and *exit-name* is the name of the user exit.

**System action:** Processing continues but no further calls are made to this exit.

**User response:** To obtain a dump of this situation for debugging purposes, allocate DDname IDITRACE to anything other than DUMMY and a dump data set to the abending job using, for example, the following JCL statements:

```
//IDITRACE DD SYSOUT=*
//SYSMDUMP DD DISP=SHR,DSN=my.dumpdsn
```

---

**IDI0038W I/O error writing to softcopy book cache: *reason***

**Explanation:** An error occurred when attempting to write to the softcopy book cache data set.

**System action:** Processing continues.

**User response:** Refer to *reason* for a possible explanation. If problems persist, delete and reallocate the data set.

---

**IDI0042W Dump data set *data-set-name* with  
timestamp *timestamp* might not match  
current history file fault entry**

**Explanation:** The dump header record timestamp *timestamp* in the dump data set name *data-set-name* was either earlier, or more than 35 minutes later, than the timestamp recorded in the current history file fault entry.

**System action:** Processing continues.

**User response:** Ensure that the dump data set has not been reused or restored with the contents of a dump from another fault.

---

IDI0043W	<b>Fetch failed for message module <i>module-name</i> - using language <i>old-language-option</i> instead of <i>new-language-option</i></b>
----------	---

**Explanation:** An attempt to bring a multicultural support message module into storage failed. The name of the load module identified in *module-name* is comprised of IDIHM followed by the Language option in effect. Either the Language option specified an unsupported language identifier, or the load module library in which the module resides was not found through the normal MVS search path.

**System action:** The current Language option setting shown in *old-language-option* is retained.

**User response:** Ensure that the Language option specifies a supported language identifier (see "Language" on page 479) and that the load module is available to Fault Analyzer through the normal MVS search path.

---

IDI0044I	<b>Current fault is a duplicate of fault ID <i>faultid</i> - the duplicate count is <i>count</i></b>
----------	--

**Explanation:** As the result of the NoDup(NORMAL(*hours*)) option in effect, Fault Analyzer determined that the current fault was a duplicate of an existing fault in the same history file. The duplicate count of the existing fault is incremented by one and is displayed as *count* in the message. Refer to "NoDup" on page 482 for the conditions that are used to determine the duplicate fault.

**System action:** Processing continues. Unless an End Processing user exit is used to change the normal behavior of duplicate fault processing, both the system dump (if any) and the history file entry is suppressed.

**User response:** None

---

IDI0046I	<b>MVS SVC dump registration skipped due to: <i>reason</i></b>
----------	--

**Explanation:** MVS SVC dump registration via the Fault Analyzer IDIXTSEL post-dump exit was not performed for the reason that is identified in *reason* as one of the following:

- **Parmlib config member Exclude option specification**

An Exclude option that is specified in the IDICNF00 configuration member matched the current fault attributes. No subsequent Include or Exclude option specifications in a user options file matched, and no attempt was made to override the exclusion by an Analysis Control (dump registration) user exit.

- **Analysis Control user exit request**

No Exclude option specification in the IDICNF00 configuration member matched the current fault attributes. Instead, an Analysis Control (dump registration) user exit requested the exclusion.

- **Abend *abend-code* during options processing**  
An abend occurred during options processing.

**System action:** No fault entry is created.

**User response:** None

---

IDI0047S	<b>IBM Fault Analyzer internal abend <i>abend-code</i></b>
----------	--

**Explanation:** Fault Analyzer terminated abnormally with the abend code shown.

This message is equivalent to the IDI0120S message, but issued under different circumstances.

**System action:** Processing terminates.

**User response:** Contact your IBM service representative.

A dump taken at the time of the IDI0047S message being issued is required for problem analysis. This dump might either be an already existing recovery fault recording dump, if available, or one can be obtained by setting the following SLIP trap before recreating the problem:

```
SL SET, ID=xxxx, MSGID=IDI0047S, A=SVCD, END
```

---

IDI0048W	<b>Incomplete minidump: Expected <i>expected-pages</i> pages, read <i>read-pages</i> pages</b>
----------	--

**Explanation:** During real-time analysis, the number of minidump pages that was expected to be saved in the history file was *expected-pages*. However, only *read-pages* was found during reanalysis.

**System action:** Processing continues.

**User response:** If *read-pages* is less than *expected-pages*, then contact your IBM service representative.

---

IDI0049S	<b>Fault ID <i>faultid</i> not found in history file</b>
----------	--

**Explanation:** A non-existing fault identifier was specified in the FaultID option.

**System action:** Processing terminates.

**User response:** Ensure that the history file that is used contains the fault ID specified in the FaultID option.

---

IDI0050S	<b>Fault reanalysis attempted without FaultID or DumpDSN option specified</b>
----------	---

**Explanation:** Reanalysis of a fault was attempted but neither the FaultID option, nor the DumpDSN option, was specified. Without either (or both) of these, Fault Analyzer cannot perform fault reanalysis.

**System action:** Processing terminates.

**User response:** Specify either the FaultID or the DumpDSN option and retry the reanalysis.

---

**IDI0052I    *count* page minidump suppressed from the fault entry being created**

**Explanation:** A minidump consisting of *count* 4K pages was suppressed by Fault Analyzer and therefore not written to the history file with the fault entry being created. However, the remainder of the fault entry is still attempted written. The suppression might be due to the maximum number of minidump pages having been exceeded, or a decision made by an invoked End Processing user exit.

**System action:** Processing continues.

**User response:** None.

---

**IDI0053I    Fault history file entry suppressed due to: *reason***

**Explanation:** No updates were made to the history file for the current fault. The reason for the suppression is provided in *reason* as one of the following:

**DUMMY history file specification**

The IDIHIST DDname was specified as DUMMY

**History file ENQ timeout**

Fault Analyzer was unable to serialize on the history file within a set period of time.

**History file access error**

If a history file open or read failure, then message IDI0016E or IDI0078E is likely to have been issued prior to this message.

**Duplicate fault or End Processing user exit**

If duplicate fault detection was the reason, then message IDI0044I is issued prior to this message. Otherwise, an End Processing user exit requested suppression.

**End Processing user exit**

During fault entry refresh processing, an End Processing user exit requested suppression.

**System action:** Processing continues.

**User response:** Unless the reason for suppression of the fault entry was that it had either been deemed a duplicate, or a user exit had requested suppression, then if the problem persists, the cause of suppression should be investigated. Check if earlier issued messages provide more information, or check if another user or job continues to hold an ENQ against the history file.

---

**IDI0055E    Fault Analyzer processing excluded because *num* meg of 31 bit storage is not currently available**

**Explanation:** Not enough above-the-line storage was available in the region for analysis to be performed.

**System action:** Processing terminates.

**User response:**

Ensure that at least *num* megabytes of above-the-line storage are available in the region to abending jobs that should be analyzed by Fault Analyzer. The necessary storage can be made available by using the JOB or EXEC JCL statement REGION<sup>21</sup> parameter. For more information, see "Storage recommendations" on page 222.

**Note:** For CICS, this entire amount of storage must be provided through the JOB or EXEC JCL statement REGION<sup>21</sup> parameter; **not** through the EDSALIM parameter.

Although Fault Analyzer was unable to perform real-time analysis due to storage constraints, recovery fault recording (see "Recovery fault recording" on page 30) is attempted in a non-CICS environment.

---

**IDI0056E    REXX environment initialization failed, IRXINIT rc=rc reason=reason**

**Explanation:** At attempt to initialize a REXX environment failed. In the message text, *rc* is the return code and *reason* is the reason code returned by the REXX initialization routine, IRXINIT. For information about the return and reason codes, see z/OS: TSO/E REXX Reference.

**System action:** Processing continues. However, no REXX exec user exits are invoked and no diagnostic information is written to the IDITRACE DDname.

**User response:** Refer to the description of return and reason codes in z/OS: TSO/E REXX Reference.

---

**IDI0057E    Dynamic load of LE failed from IDICEEDS module-name data-set-name**

**Explanation:** Either the LE runtime data set obtained from the IDICEEDS CSECT could not be opened (in which case no module name is included in the message text), or the load of a module from this data set failed. In the message text, *data-set-name* is the name of the LE runtime data set and *module-name*, if included, is the name of the load module.

**System action:** Processing terminates.

**User response:** Refer to "Identifying the LE runtime library" on page 257 for information about the IDILEDS USERMOD that can be used to modify the default LE runtime data set name when LE is not in LINKLIST and the LE runtime data set name is not CEE.SCEERUN.

---

21. Due to the MVS implementation of the REGION parameter, it might not be sufficient to simply add the required amount of storage to the value in an already specified REGION parameter.

---

<b>IDI0058W</b>	<b>Allocation of temporary work data set failed: Type=type RC=return-code Error=error-code Info=info-code</b>
-----------------	---

**Explanation:** Dynamic allocation of a temporary work data set failed.

**System action:** Processing attempts to continue without the data set.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

<b>IDI0059W</b>	<b>Logical parmlib concatenation list error: RC=return-code Reason=reason</b>
-----------------	---

**Explanation:** An attempt to use the logical parmlib list service to determine the data sets contained in the logical parmlib concatenation failed.

**System action:** Processing continues without the use of the IDICNF00 parmlib member.

**User response:** Refer to *z/OS: MVS Programming: Assembler Services Reference* for information about the return and reason codes.

---

<b>IDI0061E</b>	<b>Data set organization of history file data-set-name is not PDS or PDSE</b>
-----------------	---

**Explanation:** Fault Analyzer was invoked using an invalid format history file data set.

**System action:** Processing continues, but a fault entry cannot be written.

**User response:** Ensure that the history file is either PDS or PDSE.

---

<b>IDI0062W</b>	<b>Error in file data-set-name. reason</b>
-----------------	--

**Explanation:** An error identified in *reason* occurred when reading data set *data-set-name*.

**System action:** Processing attempts to continue.

**User response:** Message IDI0062W might be issued for a particular fault entry, or for the \$\$INDEX member. If, message IDI0062W is issued for the \$\$INDEX member due to a corrupted HI segment, then the HI segment is automatically attempted recovered from the \$\$BACKUP member on the next update of the history file, for example, when a new fault entry is written.

Ensure that the data set identified is a valid history file PDS(E), and if necessary, delete the member in error. However, care should be taken not to delete the \$\$INDEX member, unless the problem persists, as this deletion might result in the loss of duplicate fault information.

---

<b>IDI0063W</b>	<b>Deletion of file data-set-name failed. reason</b>
-----------------	--

**Explanation:** An attempt to delete data set *data-set-name* failed. The reason is identified in *reason*.

**System action:** Processing attempts to continue.

**User response:** Refer to the identified reason for the failure to delete the data set.

---

<b>IDI0064W</b>	<b>CICS Exit command failed. Response=response reason=reason</b>
-----------------	--

**Explanation:** A CICS command failed.

The following lists the possible response codes:

*Table 23. Response codes*

---

Response code	Meaning
---------------	---------

1	OK
2	Exception
3	Disaster
4	Invalid
5	Kernel error
6	Purged

The following lists the reason codes that are associated with each command:

*Table 24. Reason codes*

---

Reason code	Meaning
-------------	---------

FREEMAIN	(No applicable reason codes)
GETMAIN	
1	Insufficient storage
INQ_APPLICATION_DATA	
1	DPL program
2	No transaction environment
3	Transaction domain error
4	Invalid function
5	Abend
6	Loop
7	Inquiry failed

INQUIRE_TRANSACTION	
---------------------	--

1	No transaction environment
3	Buffer too small
7	Invalid transaction token
37	Abend
39	Loop

Table 24. Reason codes (continued)

Reason code	Meaning
<b>START_PURGE_PROTECTION</b>	
(No applicable reason codes)	
<b>STOP_PURGE_PROTECTION</b>	
(No applicable reason codes)	
<b>WAIT_MVS</b>	
3	Task canceled
4	Timed out

**System action:** Processing attempts to continue.

**User response:** Refer to the identified reason for the failure.

---

**IDI0065W CICS Exit action failed. Return code=rc**

**Explanation:** An action performed in the CICS invocation exit failed.

**System action:** Processing attempts to continue.

**User response:** Refer to the identified action and return code for the failure.

---

**IDI0066I CICS Fast No Dup processing for task task-number found duplicate for program-name abend-code fault-id**

**Explanation:** As the result of the NoDup(CICSFAST(...)) option in effect, Fault Analyzer determined that a CICS transaction abend was a duplicate of a previous CICS transaction abend. Refer to "NoDup" on page 482 for the conditions that are used to determine the duplicate fault.

In the message text, *fault-id* identifies the fault ID of which the current fault was deemed to be a duplicate.

**System action:** Processing continues without creating a history file entry for the abending CICS transaction.

**User response:** None.

---

**IDI0068I Fault Analyzer SubTask for token has been executing for num minutes.**

**Explanation:** This message occurs for CICS if a task abend has been running the Fault Analyzer SubTask for more than 10 minutes and a new CICS task abend is also needing to do Fault Analyzer SubTask Processing. The condition could occur if CPU utilization on the system is very high preventing the analysis from occurring in a reasonable time, or it could indicate a problem with Fault Analyzer SubTask processing.

**System action:** Processing continues.

**User response:** None.

---

**IDI0069E command-name failed. Resp=response-code field=value**

**Explanation:** An unknown error has occurred.

**System action:** Processing attempts to continue.

**User response:** Contact your IBM service representative and, if possible, send the joblog.

---

**IDI0071I function-name of program program-name completed successfully**

**Explanation:** This message is a confirmation that the command that was issued has completed.

**System action:** Processing terminates.

**User response:** None.

---

**IDI0072I Program program-name is status**

**Explanation:** This message provides the current exit program name and status.

**System action:** Processing continues.

**User response:** None.

---

**IDI0073I Usage: CFA <Install|Uninstall>**

**Explanation:** The arguments passed to the CFA transaction were incorrect. The message provides the user with a brief reminder of the correct usage.

**System action:** Processing continues.

**User response:** None.

---

**IDI0074E User not authorized to issue command command-name**

**Explanation:** The user that is associated with the issuing task is not authorized to use this command or is not authorized to access resources in the way required by this command.

**System action:** The attempted command was not successful.

**User response:** Ensure that the necessary command authority is provided.

---

**IDI0075E IDIXCCEE CICS exit must abort. CICS SOS pending.**

**Explanation:** Not enough storage was available to initiate analysis. This lack of storage might happen during times of high CICS system activity or when multiple CICS transaction abends are being analyzed concurrently by Fault Analyzer.

**System action:** Processing terminates.

**User response:** If the problem did not occur due to high CICS system activity or the analysis of multiple

## IDI0076E • IDI0084E

concurrent CICS transaction abends, try increasing the CICS DSA size.

---

### IDI0076E A Fault Analyzer CSVINFO MIPR routine abend occurred

**Explanation:** An abend occurred during the processing of a CSVINFO macro to retrieve information about a load module.

**System action:** CSVINFO processing terminates for the load module.

**User response:** Contact your IBM service representative.

---

### IDI0077I Fault Analyzer internal diagnostic: *text*

**Explanation:** This message is an internal Fault Analyzer message used to display specific diagnostic information.

**System action:** Processing continues.

**User response:** None.

---

### IDI0078E Open of fault history file *data-set-name* failed because: *reason*

**Explanation:** The history file that was identified by *data-set-name* could not be opened for the reason that is identified in *reason*.

If *reason* is shown as "DDfopenw() rc=X'800sssrc'", then this reason indicates an abend during open processing, where:

*sss* The hexadecimal MVS system abend code.

*rc* The associated hexadecimal reason code.

**System action:** A fault entry might not have been written to the history file.

**User response:** Determine the reason for the open failure and resubmit the job. Check that the attributes for the file definition are VB LRECL 10000 PDS(E). If the history file is a PDSE that is shared across a sysplex, then refer to "Sharing of history files across a sysplex" on page 272 for more information.

---

### IDI0081I IEWBIND unusual condition *function module-name* *rc=reason-code*

**Explanation:** An error occurred while using the Binder.

If *reason-code* is 83000526, then this reason indicates an unusual condition encountered for an input module. This condition is often the case when using a third-party product which adds its own management information to a load module. To suppress this message, use Quiet(IDI0081I).

**System action:** Processing continues.

**User response:** The value in *reason-code* is the Binder API reason code. The Binder API reason codes can be found in *z/OS: MVS Program Management: Advanced Facilities*.

---

### IDI0082E DB2 *subsys-id* Call Level Interface error: *reason*

**Explanation:** An attempt to obtain DB2 information through the DB2 Call Level Interface (CLI) failed for the DB2 subsystem identified by *subsys-id*. Refer to *reason* for details about the error.

**System action:** Processing continues but analysis might be incomplete.

**User response:** Ensure that the DB2 Call Level Interface is installed and required setup has been performed. An application plan to bind DBRMs to packages must exist. A default application plan can be created using the sample job in member DSNTIJCL of the DB2 SDSNSAMP data set (for further information, refer to *DB2 for z/OS: ODBC Guide and Reference*).

If *reason* is "SQL return code -1 for SQLAllocConnect to DB2 system *system-name*", then the problem is likely to be resolved by starting the Fault Analyzer IDIS subsystem as explained in Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239.

Included in the message *reason* is normally the result of calling the SQLError ODBC function, which provides more detailed information about the error. Of particular interest is the 8-digit hexadecimal reason code following the REASON= keyword (for example, REASON=00f30034). The explanation for this reason code can be found in *DB2 for z/OS: Codes*

---

### IDI0083E Fault Analyzer SVC error: *reason*

**Explanation:** An error occurred during the Fault Analyzer SVC execution. Refer to *reason* for details about the error.

Intermittent occurrences of this message can be expected since Fault Analyzer traverses subsystem address spaces without holding any locks.

**System action:** Processing continues but analysis might be incomplete.

**User response:** Contact your IBM service representative if the problem persists. If *reason* indicates an abend during the Fault Analyzer SVC execution, then an SVC dump should have been written. Please retain this dump for diagnostic purposes.

---

### IDI0084E Fault Analyzer Exit and Main task level mismatch

**Explanation:** An invocation exit was found to be at a different version or maintenance level than the main Fault Analyzer module, IDIDA.

**System action:** Processing continues but errors might occur.

**User response:** Ensure that all Fault Analyzer modules used are at the same version and maintenance level.

---

**IDI0085E Fault Analyzer purged, detaching subtask TCB**

**Explanation:** An abending CICS task was running Fault Analyzer when the task was purged by the operator. Fault Analyzer is detaching its analysis subtask TCB.

**System action:** Processing terminates.

**User response:** None.

---

**IDI0086E Fault Analyzer processing excluded because *size* k of 24 bit storage is not currently available**

**Explanation:** Not enough below-the-line storage was available for analysis to be performed. *size* indicates the amount of storage required in kilobytes.

**System action:** Fault analysis is excluded for the current fault.

**User response:** Ensure that sufficient below-the-line storage is available. For more information, see "Storage recommendations" on page 222.

Although Fault Analyzer was unable to perform real-time analysis due to storage constraints, recovery fault recording (see "Recovery fault recording" on page 30) is attempted in a non-CICS environment.

---

**IDI0087I *size* Meg of 31 bit storage could be provided by SETPROG LPA,ADD,MOD(*module-list*),DSN=LNKLST**

**Explanation:** When analysis could not be performed due to insufficient above-the-line storage, and all required modules of significant size have not yet been loaded into LPA, this message is issued to indicate the region size space savings that could be achieved if loading these modules into LPA. For more information, see "Storage recommendations" on page 222.

Message IDI0055E precedes this message to indicate the required above-the-line region size that could not be obtained.

**System action:** Processing terminates.

**User response:** Ensure that the amount of above-the-line storage indicated in message IDI0055E is made available. By issuing the SETPROG command provided in the text for this message, the number of megabytes indicated in *size* can be made available.

---

**IDI0088E Fault Analyzer captured IEWBIND abend *abend-code*, processing continues**

**Explanation:** While attempting to map CSECTs in a load module, Fault Analyzer invoked the IBM Binder program which abnormally terminated with the abend code indicated.

The abend code is in the format *xxxxyy*, where *xxx* is a 3-digit hexadecimal system abend code and *yyy* is the 3-digit hexadecimal representation of a 4-digit decimal user abend code.

**System action:** Processing continues, but module information might be incomplete.

**User response:** Determine the reason for the Binder failure.

---

**IDI0089I Subsystem *subsystem-name* RC=*rc*, Rsn=*rsn* *description***

**Explanation:** This message is issued when an unexpected condition occurred in the Fault Analyzer IDIS subsystem interface.

Included in the message *description* might be the result of calling the SQLError ODBC function, which provides more detailed information about the error. Of particular interest is the 8-digit hexadecimal reason code following the REASON= keyword (for example, REASON=00f30034). The explanation for this reason code can be found in *DB2 for z/OS: Codes*

**System action:** Processing continues, but the analysis might be incomplete.

**User response:** See if another message, such as IDI0082E, was also issued, which might help explain the situation.

---

**IDI0090E Execution of REXX exec *exec-name* failed, IRXEXEC rc=*return-code***

**Explanation:** This message is issued when the execution of a REXX user exit was unsuccessful.

**System action:** Processing continues although the execution of the exit failed.

**User response:** Using IDITRACE, look for REXX messages explaining the problem. Refer to *z/OS: TSO/REXX Reference* for explanation of the return code from IRXEXEC. For *rc*=20, check that the exec shown in *exec-name* is available in the IDIEXEC concatenation.

---

**IDI0091S GETMAIN failed in IDIXTSEL initial entry**

**Explanation:** Not enough storage was available for execution of the Fault Analyzer dump registration exit, IDIXTSEL.

**System action:** Dump registration processing is terminated.

**User response:** None.

---

**IDI0092S    *condition exceeded, the subtask is canceled***

**Explanation:** In the message text, *condition* might be one of the following:

**Short term interval**

Environment checking did not complete within the imposed time limit. This check is performed prior to commencing the real-time fault analysis, and the time limit having been exceeded is likely indicative of environmental problems that might otherwise have caused significant delays or hangs during the analysis.

**Time for IDIDA execution**

Real-time analysis did not complete within the maximum amount of time permitted. The time limit is automatically adjusted based on elapsed time at various checkpoints during the analysis.

**System action:** Processing is terminated.

**User response:** Contact your IBM service representative if the problem persists.

A dump taken at the time of the IDI0092S message being issued is required for problem analysis. This dump might either be an already existing recovery fault recording dump, if available, or one can be obtained by setting the following SLIP trap before recreating the problem:

SL SET, ID=xxxx,MSGID=IDI0092S,A=SVCD,END

To disable the Fault Analyzer wait and loop protection feature for other than short term interval conditions (see "Short term interval" above), the NoLoopProtection option can be used. For details, see "LoopProtection" on page 481.

---

**IDI0093W    *Binder processing stopped because num meg of 31 bit storage is not currently available***

**Explanation:** Not enough above-the-line storage is available for successful execution of the Binder.

**System action:** Binder processing is terminated, but Fault Analyzer analysis continues.

**User response:** If possible, increase the available 31-bit region size and retry.

---

**IDI0094W    *Binder processing stopped because num k of 24 bit storage is not currently available***

**Explanation:** Not enough below-the-line storage is available for successful execution of the Binder.

**System action:** Binder processing is terminated, but

Fault Analyzer analysis continues.

**User response:** If possible, increase the available 24-bit region size and retry.

---

**IDI0095W    *Unexpected condition found in source-location description***

**Explanation:** A condition was encountered which should be brought to the attention of your IBM service representative.

**System action:** Processing continues.

**User response:** Contact your IBM service representative.

---

**IDI0096S    *CICS Task *task-id* was force purged by an operator while *state* fault analysis***

**Explanation:** Analysis of a CICS transaction fault has been terminated by operator intervention.

In the message text, *task-id* identifies the task that was force purged, and *state* might be either "waiting for" or "performing" to indicate the fault analysis activity that was terminated.

**System action:** Processing terminates.

**User response:** None.

---

**IDI0097W    *Unsupported REXX execution environment detected - no REXX services available***

**Explanation:** The protection key of the job-step TCB was found to be set to a key other than 8. Since this environment is an environment which is not supported by REXX, then no REXX services are available during the fault analysis. This lack of services includes any calls to REXX user exits and all output to the IDITRACE DDname.

**System action:** Processing continues without REXX services.

**User response:** None.

---

**IDI0098S    *Exit *exit-name* NOT installed, Fault Analyzer SVC not found***

**Explanation:** The Fault Analyzer invocation exit, identified in *exit-name*), was attempted installed. However, since the Fault Analyzer SVC has not yet been installed, processing is terminated.

**System action:** Processing terminates.

**User response:** Install the Fault Analyzer SVC (see step 6 on page 232) and retry.

---

**IDI0099S** *transaction-id* transaction cannot be used,  
Fault Analyzer SVC not found

**Explanation:** The Fault Analyzer CFA transaction (which might be called by another name, identified in *transaction-id*) was attempted used. However, since the Fault Analyzer SVC has not yet been installed, processing is terminated.

**System action:** Processing terminates.

**User response:** Install the Fault Analyzer SVC (see step 6 on page 232) and retry.

---

**IDI0100S** Fault Analyzer not available, Fault Analyzer SVC not found

**Explanation:** Fault Analyzer was attempted invoked without the Fault Analyzer SVC having been installed.

**System action:** Processing terminates.

**User response:** Install the Fault Analyzer SVC (see step 6 on page 232) and retry.

---

**IDI0101I** Fault Analyzer processing skipped due to CICSDUMPTABLEEXCLUDE option.  
Abend code *abend-code*

**Explanation:** Analysis of a CICS transaction fault has been excluded as the result of using the CICSDumpTableExclude option. For details of this option, see "CICSDumpTableExclude" on page 456.

**System action:** Processing terminates.

**User response:** None.

---

**IDI0102S** Fault Analyzer processing terminated due to inappropriate execution environment

**Explanation:** An abend occurred while checking the execution environment prior to commencing fault analysis.

**System action:** Processing terminates.

**User response:** Contact your IBM service representative if the problem persists.

---

**IDI0103I** Binder processing of member *member-name* from data-set-name received a 'not found' condition.

**Explanation:** A call to the MVS Binder program was unsuccessful.

**System action:** Processing continues, but analysis might be incomplete.

**User response:** Contact your IBM service representative if the problem persists.

---

**IDI0104S** The storage prior and/or after the IDINDFUE work area has been overwritten - processing terminated.

**Explanation:** When Fault Analyzer invokes the IDINDFUE program exit, it passes a 256 byte work area. On return from IDINDFUE, Fault Analyzer has detected that the storage surrounding this work area has been overwritten.

**System action:** Fault Analyzer processing is terminated immediately.

**User response:** Investigate the IDINDFUE program exit code to determine why the storage has been overwritten.

---

**IDI0105S** *module-name:line-number* Storage allocation for dec-count (X'hex-count') bytes failed - processing terminated

**Explanation:** An invalid request for storage has occurred. This message is similar to message IDI0005S, but is used for all storage allocation requests involving a negative length.

**System action:** Processing terminates.

**User response:** Contact your IBM service representative.

A dump taken at the time of the IDI0105S message being issued is required for problem analysis. This dump might either be an already existing recovery fault recording dump, if available, or one can be obtained by setting the following SLIP trap before recreating the problem:

SL SET, ID=xxxx,MSGID=IDI0105S,A=SVCD,END

---

**IDI0106E** ENQ timed out for *major-name* *minor-name* held by *jobname* on *system-name*

**Explanation:** The time limit set for an ENQ against the history file, prior to creating a new fault entry, was exceeded. The time limit is approximately 1 minute.

The ENQ resource was held by the job *jobname* on system *system-name*.

Although a fault entry is not created, an analysis report is still written to IDIREPRT.

**System action:** Processing continues.

**User response:** Determine the reason why the ENQ could not be satisfied. This reason might be a TSO/ISPF user who is editing the history file member.

---

**IDI0107I** *transaction-id* date time exit status

**Explanation:** This message is issued whenever the Fault Analyzer Control Transaction (*transaction-id*) is used to alter the status of one or more of the CICS exits (identified by *exit*): XPCABND, XDUREQ, LE Exit, or

SDUMP Exit. The *status* field shows either Uninstalled or Installed.

**System action:** Processing continues.

**User response:** None.

---

**IDI0108I**    **The IDIS subsystem will not process *data-set-name* because it is not a PDSE history file.**

**Explanation:** This message is issued by the Fault Analyzer IDIS subsystem when PARM='UPDINDEX' is used and a history file, which is not a PDSE, is attempted opened. In the message text, *data-set-name* identifies the history file that cannot be cached in the IDIS subsystem.

For more information about IDIS subsystem caching, see "Caching of history file \$\$INDEX data" on page 240.

**System action:** Processing continues, but all I/O to the history file \$\$INDEX member is performed by the requester (real-time analysis, batch or interactive reanalysis, or Fault Analyzer ISPF interface).

**User response:** If you want to use the potential performance improvements that can be realized by having the Fault Analyzer IDIS subsystem manage the history file \$\$INDEX member, then a PDSE history file must be used.

---

**IDI0109E**    **IDI0109E PC recovery entered psw  
program-status-word abend=abend-word**

**Explanation:** This message is issued when an error has occurred in the Fault Analyzer subsystem program call interface.

Immediately following this message are usually extra messages that provide more detailed information about the error. For example:

```
IDISSPC PC recovery IDISSPC=16703250 IDISAMAN=16700000
IDISSPC R0-R3    00000001 171EAE20 80FF28FE 00000002
IDISSPC R4-R7    008A6770 008A69E0 008FD0C8 00F7DA00
IDISSPC R8-R11   00000000 00FF24AC 00000C80 00F7DA00
IDISSPC R12-R15  00000000 00000000 80FF2A2A 808A69E0
IDISSPC AR0-AR3  00000000 00000000 00000000 00000000
IDISSPC AR4-AR7  00000000 00000001 00000000 00000000
IDISSPC AR8-AR11 00000000 00000001 00000000 00000000
IDISSPC AR12-AR15 00000000 00000000 00000000 00000000
```

**System action:** Processing continues, but the subsystem service that failed might cause unexpected results.

If it is a S102 abend, then this problem can happen if the job or task that was requesting services from the Fault Analyzer subsystem was canceled before the subsystem request had completed.

**User response:** Contact your IBM service representative if the problem persists.

---

**IDI0111W**    **Timer for IDI0092S has expired but the NoLoopProtection option was set, execution continues**

**Explanation:** This message is issued if the loop/wait protection feature interval timer expires, but the NoLoopProtection option is in effect.

**System action:** Processing continues.

**User response:** None.

---

**IDI0112W**    ***db2\_name* SYSIBM.SYSDBRM access time took *num\_secs* seconds, further IDIS requests for this table suspended for 30 minutes—create index on SYSIBM.SYSDBRM for improved performance**

**Explanation:** This message is issued if the IDIS subsystem attempt to query the SYSIBM.SYSDBRM DB2 catalog table for DB2 subsystem name *db2\_name* exceeded the expected maximum time indicated by the *num\_secs* value. No further attempts to access the SYSIBM.SYSDBRM table on the identified DB2 subsystem is made for 30 minutes.

**System action:** Processing continues, but some DB2-related information might be missing from the analysis report for the fault that caused this message to be issued, as well as for any subsequent faults involving the same DB2 subsystem.

**User response:** Create an index on SYSIBM.SYSDBRM for improved performance. For details, see "Improving Fault Analyzer DB2 performance" on page 317.

---

**IDI0113W**    **Subtask *tcb-addr*(*description*) return ECB=*ecb-value* during problem reanalysis under CICS**

**Explanation:** An error occurred while performing interactive reanalysis under CICS. This error might either be due to incomplete setup of the interactive reanalysis component under CICS (see "Performing interactive reanalysis under CICS" on page 206), or because an abend occurred.

**System action:** Interactive reanalysis terminated.

**User response:** Check for other messages that might help explain the problem and contact your IBM service representative if the problem persists.

---

**IDI0114W**    **XMEM POST ABEND S102, requester termination**

**Explanation:** This problem can happen if the job or task that was requesting services from the Fault Analyzer subsystem was canceled before the subsystem request had completed.

**System action:** The analysis has already been canceled.

**User response:** Contact your IBM service representative if the problem persists.

---

**IDI0115E LE enclave abend *system-abend-code user-abend-code*, execution continues**

**Explanation:** An abend occurred during the fault analysis.

**System action:** Processing continues.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0116E IDILANGX abend *abend-code*, execution continues**

**Explanation:** An abend occurred in the IDILANGX program.

**System action:** Processing continues, but source-level information might be incomplete.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0117E ABEND<*abend-code*> during return POST, request from task *jobname job-id* may have been canceled**

**Explanation:** An abend occurred in the Fault Analyzer subsystem POST processing. This abend might have been caused by a CANCEL of the request from the task identified by *jobname* and *job-id*.

**System action:** Processing continues.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0118W CICS task *task-number* abend *abend-code* analysis skipped due to max waiting exceeded.  
CICSFAtasks(*max\_slots,max\_waits*)**

**Explanation:** The maximum number of faults allowed to be queued for analysis had already been reached when another fault occurred. This limit is controlled by the CICSFAtasks suboption of the DeferredReport option, and the current values in effect are provided in the message text.

For information about the DeferredReport option, see “DeferredReport” on page 463.

**System action:** Analysis is skipped, and normal CICS transaction dump analysis needs to be performed (that is, not using Fault Analyzer).

**User response:** Unless the maximum values are already in effect, the CICSFAtasks suboption of the DeferredReport option can be used to permit more parallel execution tasks to be used, or to increase the number of faults that are allowed to be waiting for analysis.

---

**IDI0119E IDIS subsystem request *server-id resource-id* by *jobname job-id* has hung and will be canceled**

**Explanation:** The Fault Analyzer IDIS subsystem has detected that a request has not terminated as expected.

**System action:** The request is canceled.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0120S IBM Fault Analyzer internal abend *system-abend-code user-abend-code***

**Explanation:** Fault Analyzer terminated abnormally with either system abend code *system-abend-code* or user abend code *user-abend-code*.

This message is equivalent to the IDI0047S message, but issued under different circumstances.

**System action:** Processing terminates.

**User response:** Check for the occurrence of previously issued severe (S-level) Fault Analyzer messages that might explain the reason for the problem (for example, message IDI0005S which indicates a storage shortage problem). If no prior S-level messages were found, contact your IBM service representative.

---

**IDI0121I ImageFast NoDup processing found duplicate of *ims\_pgm fault\_id histfile***

**Explanation:** Fault Analyzer determined that the current fault was an IMS NoDup(ImageFast(...)) duplicate. (For information about IMS NoDup(ImageFast(...)) duplicate detection, see “NoDup” on page 482.) The earlier occurring fault, of which the current fault was deemed to be a duplicate, is identified by the IMS program name (*ims\_pgm*), the assigned fault ID (*fault\_id*), and the history file data set name (*histfile*).

**System action:** Processing continues. No fault analysis is performed, but the duplicate count of the earlier occurring fault is incremented, and the duplicate details recorded for later viewing with the Fault Analyzer ISPF interface (see “Viewing the fault entry duplicate history” on page 87).

**User response:** None.

---

**IDI0122W User exit IDIALLOC command failed:  
DD=*ddname* DSN=*data-set-name*  
RC=*return-code* Error=*error-code*  
Info=*info-code***

**Explanation:** A REXX user exit issued an IDIALLOC command to dynamically allocate a data set, but the allocation failed.

**System action:** Processing continues without allocation of the data set identified in *data-set-name*.

**User response:** Refer to *z/OS: MVS Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

---

**IDI0123S Processing of abend *abend-code* terminated due to unsupported execution environment: *reason***

**Explanation:** Prior to, or during analysis of the abend shown in *abend-code*, Fault Analyzer determined that processing could not proceed for one of the following reasons:

- **SUB=MSTR address space**

The address space was started with SUB=MSTR and does therefore not permit SYSOUT allocations.

- **Not full function address space**

The current address space was found to be a limited function address space, which does not support required system services, such as allocation of data sets.

- **System address space**

The started task was a known system address space for which Fault Analyzer analysis is not appropriate and likely to fail.

- **Abend in system task**

The abend occurred under an RB for a system task, such as the IEFIIIC initiator TCB.

- **JES not available**

The primary subsystem (JES) was found to be not available.

- **Fault Analyzer invoked during JES shutdown**

Fault Analyzer determined that JES became unavailable during the processing of an abend.

- **Dynamic Allocation not available**

Required Dynamic Allocation services (SVC 99) were found to be not available for the current task.

- **Owning Job-step TCB is abending**

If the current TCB is not the job-step TCB, and the owning job-step TCB is terminating, then load failures are likely to occur.

- **Abend in resource manager**

A resource manager was active for the current TCB.

- **Concurrent analysis in the address space**

To prevent exhausting system resources, only one TCB in a given address space is allowed to perform Fault Analyzer abend analysis at a time. When Fault Analyzer was invoked for the current abend, it was determined that another Fault Analyzer abend analysis was already active, causing the current analysis to be canceled.

- **SYSZTIOT enqueued by TCB *tcb-address***

Fault Analyzer was invoked to analyze an abend with the SYSZTIOT major name already enqueued by the TCB identified by the address in *tcb-address*. An example of when this situation might occur is

during shutdown of IMS when Fault Analyzer is invoked for a U0002 abend.

- **TCB Not protect KEY 8 or 9**

Fault Analyzer was invoked to analyze an abend for a task not running in protect key 8 (non-CICS) or 9 (CICS).

**System action:** Processing terminates.

**User response:** Real-time analysis is not possible under these conditions. Fault Analyzer might still be used to aid with the problem determination by setting a SLIP trap using the MSGID=IDI0123S and A=SVCD parameters, and then performing analysis of the resulting SVC dump by way of the Fault Analyzer ISPF interface **File** menu option 5.

---

**IDI0124E IDIS subsystem subtask *server-id* *resource-id* has terminated with abend code *abend-code***

**Explanation:** A subtask in the Fault Analyzer IDIS subsystem terminated abnormally.

**System action:** The IDIS subsystem subtask is terminated, but is restarted as necessary.

Requester processing continues, but results might be unexpected depending on the type of subsystem request that failed.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0125W IDIS subsystem subtask *server-id* *resource-id* has terminated with return code *rc***

**Explanation:** A subtask in the Fault Analyzer IDIS subsystem terminated abnormally.

**System action:** The IDIS subsystem subtask is terminated, but is restarted as necessary.

Requester processing continues, but results might be unexpected depending on the type of subsystem request that failed.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0126I Recovery fault recording fault ID *faultid* assigned in history file *history-file-name***

**Explanation:** Fault Analyzer was unable to complete normal real-time analysis, but has created a recovery fault recording fault entry *faultid* in history file *history-file-name*. Reanalysis of this fault entry should provide information about the fault that was being analyzed when Fault Analyzer terminated.

**System action:** Processing has ended.

**User response:** None

---

**IDI0127W Recovery fault recording failed for job-id.  
reason**

**Explanation:** Recovery fault recording processing failed for the job identified by *job-id* and the reason that is identified in *reason* as one of the following:

- **SDUMP suppressed, capture phase of another SVC dump was in progress.**

SDUMP failed with return code 8 and reason code 2, indicating that an SVC dump was suppressed because the capture phase of another SVC dump was in progress.

- **SDUMP DASD space or overload error.**

SDUMP failed with return code 8 and reason code 3E, indicating that SVC dump is already using the maximum amount of virtual storage (as determined by the installation, using the MAXSPACE parameter on the CHNGDUMP command) to process other dumps.

Typically, the reason for this condition is a shortage of DASD space.

- **SDUMP rc=return-code reason=reason-code error.**

SDUMP failed with return code *return-code* and reason code *reason-code*. Refer to *z/OS: MVS Programming: Authorized Assembler Services Reference, Volume 3 (LLACOPY-SDUMPX)* for an explanation of these codes.

- **RFR requires the IDIS subsystem to be functioning correctly.**

A situation occurred for which recovery fault recording would normally be performed. However, since the IDIS subsystem was not functioning correctly, recovery fault recording was not possible.

**System action:** Termination continues.

**User response:** Perform any appropriate action, depending on the reason that is identified.

To enable recovery fault recording, the IDIS subsystem must be started and be functioning.

---

**IDI0128I Recovery fault recording data set  
*data-set-name* not deleted due to  
insufficient access  
authorization—associated fault ID was  
*fault-id* in history file *hist-file* created by  
*jobname* *jobname* and user ID *user-id*  
with security server default group ID  
*group-id***

**Explanation:** A recovery fault recording fault entry was deleted and an attempt was made to also delete the associated RFR dump data set. However, due to insufficient access authorization, the RFR dump data set identified by *data-set-name* could not be deleted.

The fault entry deletion occurred due to one of the following:

- The use of the D (or DD) line command against a history file fault entry from the ISPF interface.
- The use of a DELETE control statement with the IDIUTIL batch utility.
- Automatic fault entry deletion due to the maximum number of history file fault entries set using the IDIUTIL batch utility SetMaxFaultEntries control statement having been reached, or due to an AUTO-managed PDSE history file having used up all available space in the currently allocated extents. This deletion might occur during real-time analysis, fault entry creation following interactive MVS dump analysis, or IDIUTIL batch utility IMPORT processing.

**System action:** Processing continues.

**User response:** Delete the identified RFR dump data set.

From the extra information that is provided in this message, determine if security server access authorization to RFR dump data sets of this origin should be changed to allow for their automatic deletion along with their associated fault entries. For more information, see “Recovery fault recording” on page 30.

---

**IDI0129W Recovery fault recording IEATDUMP  
failure, rc=return-code reason=reason-code  
dump=*data-set-name***

**Explanation:** An error occurred while attempting to write the IEATDUMP data set during recovery fault recording processing. In the message text, *return-code* and *reason-code* are the return and reason codes from the IEATDUMP service, while *data-set-name* is the name of the dump data set attempted written. Refer to the section on IEATDUMP return and reason codes in *z/OS: MVS Programming: Assembler Services Reference* for an explanation of the problem.

**System action:** Termination continues.

**User response:** Perform any appropriate action to resolve the IEATDUMP error.

---

**IDI0130E Response from IDIS subsystem *task-id1*  
*task-id2* not returned within 2 minutes,  
request canceled**

**Explanation:**

**System action:** Processing continues, but the canceled subsystem request might affect the expected result.

**User response:** Determine the reason why the IDIS subsystem is not responding by checking for other messages that might have been issued. Also ensure that the IDIS subsystem has not been prioritized lower than any of the abending tasks for which it might be invoked.

If missing responses are persistent, and no reason can

be determined, then please contact your IBM service representative.

---

**IDI0131W Waiting *mins* minutes for *dsn(mbr)* SPFEDIT ENQ**

---

**Explanation:** This message indicates a problem with obtaining exclusive access to the history file data set member identified by *dsn* and *mbr*. The period waited so far is indicated as a number of minutes in the *mins* value.

**System action:** The IDIS subsystem continues to wait for the required data set access.

**User response:** Determine why the identified data set member is not available for update. A possible reason is that a TSO/ISPF user is editing the member.

---

**IDI0132W MaxWaitSeconds of *seconds* exceeded for transaction *transaction (task task)*, analysis will be skipped**

---

**Explanation:** The MaxWaitSeconds value in effect for the DeferredReport(CICS(FATasks(...))) option was exceeded. For details about this option, see "DeferredReport" on page 463.

**System action:** No analysis is performed for the identified CICS transaction.

**User response:** None.

---

**IDI0133W DeferredReport option overridden due to MaxMinidumpPages(*max\_pages*) exceeded by *num\_pages* pages**

---

**Explanation:** The DeferredReport option was in effect while the MaxMinidumpPages option limit was exceeded.

**System action:** While no minidump was written to the fault entry, the analysis was still performed and a report was written to both the fault entry and to IDIREPRT.

**User response:** Ensure that the MaxMinidumpPages option setting is sufficiently high to prevent frequent occurrences of this situation.

---

**IDI0134E Fault Analyzer processing excluded because *size k* of 24 bit LSQA storage is not currently available**

---

**Explanation:** The minimum required amount of storage shown as *size* in kilobytes was not available for the below-the-line LSQA.

**System action:** Processing is terminated.

**User response:** None.

---

**IDI0135E Recovery fault recording terminating. Severe private storage shortage and no SVC dump access.**

---

**Explanation:** Not enough storage was available in the private region to perform recovery fault recording processing using the IEATDUMP dump type, and Fault Analyzer was unable to use the SDUMP dump type due to insufficient access authority. The SDUMP is written from the IDIS subsystem, and is therefore normally able to be used when there is a severe storage shortage in the abending region, which prevents the IEATDUMP from being written.

**System action:** Processing is terminated without a recovery fault recording fault entry.

**User response:** If possible, resubmit with a larger region size, or provide the necessary access authority to use the SDUMP dump type instead (see "SDUMP recovery fault recording data sets" on page 235).

---

**IDI0136W Recovery fault recording IEATDUMP not taken because NULLFILE has been selected for the DSN**

---

**Explanation:** Recovery fault recording processing was unable to write an IEATDUMP data set due to no eligible data set name having been determined.

**System action:** Processing is terminated.

**User response:** None.

---

**IDI0137W I/O Error**

---

**Explanation:** An error occurred during a data set I/O operation.

**System action:** Processing continues.

**User response:** Check for more messages related to this error.

---

**IDI0138S No minidump or MVS dump data set is available for reanalysis of history file *hist-file* fault ID *fault-id***

---

**Explanation:** Batch reanalysis was attempted of a fault entry that did not either include a minidump, or was associated with an existing MVS dump data set. Without one or both of these, reanalysis is not possible.

**System action:** Processing is terminated.

**User response:** None.

---

**IDI0140S Processing terminated due to data set open error for DDname *ddname***

---

**Explanation:** The open of a data set for a required DDname failed.

Normally, this message is preceded by one or more other messages which provide more detailed

information about the error (for example, message IDI0006E).

**System action:** Processing is terminated.

**User response:** Check options specification relating to the identified DDname.

---

**IDI0141W Please use MODIFY-STOP for the IDIS subsystem**

**Explanation:** This message is issued if the CANCEL command was used to stop the IDIS subsystem.

The correct way to stop the IDIS subsystem is to use the MODIFY or STOP command:

F *name*,STOP

or

P *name*

**System action:** The IDIS subsystem is stopped.

**User response:** None.

---

**IDI0142W Dispatch delay in IDIS subsystem (Priority=idis-priority) exceeded num seconds for jobname job-id (Priority=job-priority)**

**Explanation:** This message is issued if the IDIS subsystem response time exceeded expectations for the requester shown.

**System action:** Processing continues.

**User response:** Review the dispatch priority of the IDIS subsystem to ensure that it is not less than that of the job identified by *jobname* and *job-id* which was requesting Fault Analyzer subsystem services.

---

**IDI0143W Binder processing stopped because num k of 24 bit LSQA storage is not currently available**

**Explanation:** Not enough 24-bit LSQA storage was available to invoke the Binder.

**System action:** Processing continues, but source line information might be missing from the analysis report.

**User response:** None.

---

**IDI0144E IDIS subsystem TCB *tcb-address* detection-location **abended** *abend-code***

**Explanation:** An IDIS subsystem TCB or function abended or detected an error condition.

**System action:** Processing continues, but information might be missing from the analysis report.

**User response:** If the problem persists, contact your IBM service representative. If a fix is not available, then

set a SLIP trap as follows and provide the dump to IBM for analysis:

SL SET, ID=xxxx,MSGID=IDI0144E,A=SVCD,END

---

**IDI0145I *message-text***

**Explanation:** This message is used for all status messages that are issued by the IDIS subsystem.

Examples of these messages are:

IDI0145I IDISXCFA TCB XCF startup  
IDI0145I IDIS subsystem, IDISMAIN Started. V9R1M0  
(MVS 2009/02/03)  
IDI0145I Starting Termination.

**System action:** Processing continues.

**User response:** None.

---

**IDI0146I IDIS subsystem storage use is at percent-value percent of JCL REGION size, running number-of-tasks tasks**

**Explanation:** This message is issued by the IDIS subsystem whenever the amount of storage used exceeds 80% of the maximum available storage, as specified in the JCL REGION parameter, or imposed by default. The message is reissued at intervals while the storage usage remains high, but is not issued if the storage usage drops below 80% again.

**System action:** Processing continues.

**User response:** If this message is regularly issued by the IDIS subsystem, then it is recommended to increase the region size to avoid termination of the IDIS subsystem due to insufficient storage being available.

---

**IDI0147I Alter access to XFACILIT IDI\_SDUMP\_ACCESS is required for an SDUMP dump**

**Explanation:** This message is issued by the IDIS subsystem whenever an RFR dump is written, and it is determined that the user ID associated with the abending job does not have ALTER access to the XFACILIT IDL\_SDUMP\_ACCESS resource class.

**System action:** The RFR dump is attempted written as an IEATDUMP instead of an SDUMP.

**User response:** See “SDUMP recovery fault recording data sets” on page 235 for information about how to use SDUMPs for improved recovery fault recording performance.

---

**IDI0149W IDIMAPS *dsname* has a build YYMMDD=*build-date* but the required level is *required-date*. Execution may be incorrect.**

**Explanation:** An incorrect version of the IDIMAPS data set is being used. The data set identified by *dsname* was built on the date shown in *build-date*, which did

## IDI0150W • IDI0156W

not match the required date for the installed level of Fault Analyzer in *required-date*.

**System action:** Processing continues, but results might not be correct.

**User response:** Ensure that the data set name that is specified for the IDIMAPS DDname is correct (this name is normally specified using the DataSets option in the IDICNFxx parmlib member), and contains the current data from the SMP/E target library.

---

**IDI0150W No READ access to DDname *ddname* data set name *dsname*. This data set will not be used.**

**Explanation:** A data set provided to Fault Analyzer, either explicitly or implicitly, was found to be inaccessible due to no security server READ access.

**System action:** Processing continues, but errors might occur if the data set is critical.

**User response:** Provide the appropriate access to the data set.

---

**IDI0151W SDUMP failure reason**

**Explanation:** Recovery fault recording processing or Java analysis failed with the reason that is identified in *reason* as one of the following:

- SDUMP rc=8 rsn=2 for job *jobname*. SVC dump suppressed because capture phase of another SVC dump was in progress.
- SDUMP rc=8 rsn=3E for job *jobname*. DUMPSERV has used virtual storage MAXSPACE because of dump DASD space shortage or high activity.
- SDUMP rc=0 rsn=04 for job *jobname*. DUMPSERV could only take a partial dump. Examine associated IEA\* DUMP messages for more detail.
- SDUMP rc=*return-code* rsn=*reason-code* for job *jobname*.
- SDUMP requires the IDIS subsystem to be functioning.

See message IDI0127W (“IDI0127W” on page 551) for more information about the SDUMP return and reason codes.

This message is issued by the IDIS subsystem.

**System action:** IDIS subsystem processing continues.

**User response:** Perform any appropriate action, depending on the reason that is identified.

---

**IDI0152I Job *jobname* SDUMP requested for history-file(*fault-id*)**

**Explanation:** An SDUMP recovery fault recording dump data set was requested for the job identified by *jobname*, which is to be associated with fault ID *fault-id* in history file *history-file*.

**System action:** Processing continues.

**User response:** None.

---

**IDI0153I Binder processing terminated for member *module-name* because it was created with the LINK=NO option**

**Explanation:** A call to the Binder program for load module *module-name* failed with rc=83000505 due to the use of the LINK=NO linkedit option.

**System action:** Processing continues without binder information for the identified load module.

**User response:** None.

---

**IDI0154W Configuration-options module *module-name* found in non-authorized load library and has been ignored**

**Explanation:** The load module identified by *module-name*, which can be used to provide configuration-options for Fault Analyzer, was found in a load library which was not APF-authorized. The load module name can be either IPVOPTLM or IDIOPTLM. Because it is a requirement that this load module must be placed in an APF-authorized load library in order to be used, it has been ignored.

**System action:** Processing continues.

**User response:** Place the load module in an APF-authorized load library.

---

**IDI0155W The user ID in use for the IDIS subsystem does not have an OMVS segment with a HOME path**

**Explanation:** If an IDIJLIB DDname has not been specified in the IDIS subsystem JCL, then Fault Analyzer instead uses the IDIS subsystem user ID OMVS segment HOME path for work files. In this case, neither an IDIJLIB DDname, nor a HOME path were available.

**System action:** Processing continues, but without Java analysis support.

**User response:** Ensure that either the IDIJLIB DDname has been specified in the IDIS subsystem JCL, or that an OMVS segment HOME path exists for the IDIS subsystem user ID.

---

**IDI0156W GETMAIN of *count* bytes from jobname *job-id* address space failed *abend-code*, *idis-module-name* *history-file-name* IDIS subsystem request failed**

**Explanation:** When the IDIS subsystem was returning data, a cross memory GETMAIN for *count* bytes failed, resulting in abend *abend-code*. This abend occurred because of storage shortage in the requester address space identified by *jobname* and *job-id*.

**System action:** Processing continues.

**User response:** If possible, increase the region size of the requester address space to prevent this problem from happening in the future.

---

**IDI0157I Fault Analyzer about to deliberately abend U0777 and take RFR dump due to IDIRFRON DDname**

**Explanation:** When an IDIRFRON DD statement is used, Fault Analyzer deliberately issues abend U0777 in order to cause a recovery fault recording fault entry to be created. For more information about the use of the IDIRFRON DDname, see "Verifying the recovery fault recording set-up" on page 335.

**System action:** Processing terminates.

**User response:** None.

---

**IDI0158W IDIS subsystem requires restart with STEPLIB containing SMP/E \*.SIDIAUT2 to load DLL *dll-name***

**Explanation:** Analysis of a Java fault was attempted but failed due to the absence of data set IDI.SIDIAUT2 in the IDIS subsystem STEPLIB concatenation.

**System action:** Analysis continues but Java information is missing.

**User response:** Add IDI.SIDIAUT2 to the IDIS subsystem STEPLIB concatenation. For details, see "Starting the IDIS subsystem" on page 241.

---

**IDI0159I SDUMP requested for *job-id1* will not be taken because of high *job-id2* usage of SDUMP**

**Explanation:** The rate of Fault Analyzer recovery fault recording SDUMPs (SVC dumps) scheduled exceeded the operating system's capacity to handle these, and as a result, the current dump was not taken. In the message text, *job-id1* is the JES job ID of the job for which the SDUMP was requested, and *job-id2* is one of the following:

- The same as *job-id1*, if the SDUMP rate threshold for the current job has been exceeded.
- "System", if the threshold for all jobs combined has been exceeded.

**System action:** The recovery fault recording SDUMP is not taken.

**User response:** Contact your IBM service representative to determine the reason why a high rate of Fault Analyzer recovery fault recording SDUMPs were requested, and provide examples of the SDUMPs that were taken shortly before this message was issued for analysis.

---

**IDI0160I History file *history-file* I/O error recovery successful**

**Explanation:** Following an I/O error indicated by message IDI0033E, Fault Analyzer was able to reclaim sufficient space in the history file identified by *history-file* to permit the subsequent successful rewriting of the fault entry.

**System action:** Processing continues normally.

**User response:** None.

---

**IDI0161W History file *history-file* I/O error recovery failed: *reason***

**Explanation:** An attempt to recover from an I/O error on the history file identified by *history-file* failed for the reason that is identified by *reason* as one of the following:

- **History file is not a PDSE**  
I/O error recovery is not available for PDS history files.
- **History file contains 25 or less fault entries**  
As for AUTO-space management, fault entries are only implicitly deleted if the history file contains more than 25 fault entries.
- **IGWFAMS error**  
Message IDI0095W is issued immediately prior to this message with error-specific information.
- **Unable to provide the required space**  
No more fault entries could be deleted, but the required amount of space had not yet been made available. This lack of required space might be due to fault entries being locked.
- **Recursive I/O error**  
An I/O error occurred during the recovery of an earlier I/O error.

**System action:** Processing continues, but a fault entry is not written to the history file.

**User response:** One or more of the following might be appropriate:

- Reallocate the history file with more space.
- Change the history file space management setting using the IDIUTIL SetMaxFaultEntries or SetMinFaultEntries control statement.
- Unlock locked fault entries so they can be implicitly deleted.

---

**IDI0162I MVS dump taken to extract Java information**

**Explanation:** An MVS dump has been taken For the purpose of performing asynchronous extraction of Java information for the current fault.

**System action:** Processing continues.

**User response:** None.

---

**IDI0164I Fault ID *fault-id* created in history file *history-file* due to *reason***

**Explanation:** The batch analysis of an MVS dump data set caused the creation of a new fault entry *fault-id* in history file *history-file* due to *reason*.

The possible values for *reason* are:

- GenerateSavedReport option
- IDIRRegisterFaultEntry command

**System action:** Processing continues.

**User response:** None.

---

**IDI0165A *ddname* updates available--restart IDIS subsystem with //IDIDOC2 DD statement for SIDIDOC2 data set**

**Explanation:** During start-up of the IDIS subsystem, it was determined that one or more updates were available for the VSAM KSDS message and abend code explanation repository, identified by *ddname* as either IDIVSENU, IDIVSJPN or IDIVSKOR. However, no //IDIDOC2 DD statement had been provided to identify the IDI.SIDIDOC2 data set containing the updates.

**System action:** Processing continues without updating the VSAM KSDS.

**User response:**

1. Add a DD statement in the IDIS subsystem JCL as follows for the IDI.SIDIDOC2 data set containing the VSAM KSDS updates:  
//IDIDOC2 DD DISP=SHR,DSN=IDI.SIDIDOC2
2. Restart the IDIS subsystem.

---

**IDI0166E Error processing *ddname* update *member*: *reason***

**Explanation:** An error occurred while attempting to update the VSAM KSDS message and abend code explanation repository. In the message text:

*ddname* Identifies the DDname as either IDIVSENU, IDIVSJPN or IDIVSKOR.

*member* The name of the update member in data set IDI.SIDIDOC2.

*reason* A description of the error which occurred.

**System action:** Processing continues, but the VSAM KSDS update is incomplete and is attempted again next time the IDIS subsystem is started.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0167I Models processing for CICS release returned *count* offsets**

**Explanation:** Under CICS, Fault Analyzer needs certain release dependant field offset values. These offsets are determined when Fault Analyzer is installed during CICS startup, and this message gives a count of the number of offsets found.

**System action:** Processing continues.

**User response:** None.

---

**IDI0168E Models processing for CICS release returned 0 offsets**

**Explanation:** Under CICS, Fault Analyzer needs certain release dependant field offset values. This message is issued when the processing has failed to determine any offset information. This failure prevents Fault Analyzer operation under CICS.

**System action:** Fault Analyzer operation under CICS is not possible without the offset information.

**User response:** The offset information is extracted from the SIDIMAPS data set member IDIMCICS. Ensure that the IDIMAPS DataSets option has been specified correctly. If the problem persists, then set a SLIP trap on this message number and contact your IBM service representative.

---

**IDI0169E Module *module-name* loaded from *library-name* is not APF authorized or the concatenation is not APF authorized**

**Explanation:** It is a requirement that the Fault Analyzer load module, identified by *module-name*, is executed from an APF-authorized load library. In this case, the load module was executed from the load library identified by *library-name*, which was either not APF-authorized, or was included in a JOBLIB or STEPLIB concatenation along with one or more other load libraries which were not APF-authorized.

**System action:** Processing terminates.

**User response:** Ensure *library-name* is APF-authorized and not included in a concatenation with other load libraries which are not APF-authorized.

---

**IDI0170W Unable to update *data-set-name* due to no UPDATE access**

**Explanation:** This message is issued by the Fault Analyzer IDIS subsystem if updates are available for the message and abend code explanation repository data set identified by *data-set-name* but the IDIS subsystem does not have UPDATE access to this data set. The updates might be to correct formatting issues or typos in existing explanations, or to add new messages or abend codes.

**System action:** Processing continues without performing the update.

**User response:** To perform the update, first grant UPDATE access to *data-set-name* by the Fault Analyzer IDIS subsystem, then stop and restart the IDIS subsystem.

---

**IDI0171W Waiting count minutes for history-file-name PDSE cross-system SHARING contention**

**Explanation:** This message is issued by the Fault Analyzer IDIS subsystem if access to a history file is taking longer than one minute to be obtained, and is issued every minute thereafter until the access has been obtained.

**System action:** Processing of the history file is suspended until the access is obtained.

**User response:** If the problem persists, determine who is holding an ENQ on the history file using the SPFEDIT major name.

---

**IDI0172I Fault Analyzer NameToken anchor built at storage-addr by TCB tcb-addr**

**Explanation:** This message is issued during Fault Analyzer initialization under CICS. Its purpose is to provide diagnostic aid for IBM in case of errors.

**System action:** Processing continues normally.

**User response:** None.

---

**IDI0173I Fault Analyzer system DUMP call already issued for Fault Entry fault-id**

**Explanation:** During recovery fault recording (RFR) processing, the fault entry identified by *fault-id* was found to already have had a system dump call issued for it. Hence, an extra system dump call is not issued.

**System action:** Processing continues normally.

**User response:** None.

---

**IDI0174I Fault Analyzer java DUMP fault entry written to history-file1(fault-id) because IDIS subsystem cannot write to history-file2**

**Explanation:** The Fault Analyzer IDIS subsystem was unable to write a Java dump fault entry to history file *history-file2* because of insufficient access authorization. Instead, the fault entry *fault-id* was written to the default history file *history-file1*.

**System action:** Processing continues normally.

**User response:** None.

---

**IDI0175I Fault Analyzer DUMP analysis will use dump-dsn for history-file(fault-id)**

**Explanation:** This message is issued by the IDIS subsystem when the dump data set identified by *dump-dsn* has been determined to belong to the fault entry identified by *history-file(fault-id)*. The fault entry is updated to add the associated dump data set name.

**System action:** Processing continues normally.

**User response:** None.

---

**IDI0177E Java DTFJ processing failed for history-file(fault-id)**

**Explanation:** Expected Java information was not found during reanalysis of fault entry *fault-id* in history file *history-file*. This message suggests that Java DTFJ processing failed to complete successfully.

**System action:** Processing continues normally, but without Java information.

**User response:** Determine the reason why DTFJ processing failed. Ensure the MVS post-dump exit IDIXTSEL is installed (for details, see "Installing the MVS post-dump exit IDIXTSEL" on page 313) and the IDIS subsystem is started (for details, see Chapter 14, "Using the Fault Analyzer IDIS subsystem," on page 239).

---

**IDI0178E Fault Analyzer startup has hung. RFR dump initiated to capture the abend.**

**Explanation:** A hang was detected while attempting to perform real-time analysis.

**System action:** The real-time analysis is terminated and a recovery fault recording fault entry is written instead.

**User response:** If the problem persists, contact your IBM service representative.

---

**IDI0179W Process error trying to force side-file-type side-file use for program-name. The side-file will not be used.**

**Explanation:** The "Listing/Side File Mismatch" prompt, "ENTER" key action could not successfully process the side file.

**System action:** Processing continues without compiler listing or side file support for the program.

**User response:** If required, check the compile date and the reasons listed for the mismatch, and locate a compiler listing or side file that is a better match for the program.

---

**IDI0180I**    **Fault Analyzer processing skipped due to CICSDUMPTABLEEXCLUDE(CheckMaxCurr). Abend code *abend-code*.**

**Explanation:** A CICS transaction dump table entry exists for *abend-code* and the current dump count exceeds the maximum setting. Due to the CICSDUMPTABLEEXCLUDE(CheckMaxCurr) option being specified, Fault Analyzer analysis is skipped.

**System action:** Processing terminates.

**User response:** None

---

**IDI0181W**    **IDIS now stopping and restarting tasks to conserve storage. Increase REGION size and restart.**

**Explanation:** Fault Analyzer has detected a Short on Storage condition within the subsystem. The subsystem is stopping and restarting the IDIS DB2 subtasks when necessary to try to alleviate the condition.

**System action:** Processing continues.

**User response:** If this message is regularly issued by the IDIS subsystem, then it is recommended to increase the region size to avoid termination of the IDIS subsystem due to insufficient storage being available.

---

**IDI0182I**    **Fault Analyzer subsystem error from IDISAREQ: *error\_message***

**Explanation:** A message was passed back from the Fault Analyzer IDIS subsystem due to an unexpected condition. The passed-back message in *error\_message* usually includes a separate message ID.

**System action:** Processing continues.

**User response:** Refer to the message ID in *error\_message* for the problem that occurred in the Fault Analyzer IDIS subsystem call.

---

## IDILANGX messages

Messages prefixed by IDISF\* are issued by the IDILANGX program, which is used internally by Fault Analyzer or invoked by the user when creating side files. These are documented in *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

---

## **Part 4. Appendixes**



---

## **Appendix A. Support resources and problem solving information**

This section shows you how to quickly locate information to help answer your questions and solve your problems. If you have to call IBM support, this section provides information that you need to provide to the IBM service representative to help diagnose and resolve the problem.

For a comprehensive multimedia overview of IBM software support resources, see the IBM Education Assistant presentation “IBM Software Support Resources for System z Enterprise Development Tools and Compilers products” at <http://publib.boulder.ibm.com/infocenter/iedusst/stgv1r0/index.jsp?topic=/com.ibm.iea.debugt/debugt/6.1z/TrainingEducation/SupportInfoADTools/player.html>.

- “Searching knowledge bases”
- “Getting fixes” on page 563
- “Subscribing to support updates” on page 563
- “Contacting IBM Support” on page 564

---

### **Searching knowledge bases**

You can search the available knowledge bases to determine whether your problem was already encountered and is already documented.

- “Searching the information center”
- “Searching product support documents”

### **Searching the information center**

You can find this publication and documentation for many other products in the IBM System z Enterprise Development Tools & Compilers information center at <http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp>. Using the information center, you can search product documentation in a variety of ways. You can search across the documentation for multiple products, search across a subset of the product documentation that you specify, or search a specific set of topics that you specify within a document. Search terms can include exact words or phrases, wild cards, and Boolean operators.

To learn more about how to use the search facility provided in the IBM System z Enterprise Development Tools & Compilers information center, you can view the multimedia presentation at <http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp?topic=/com.ibm.help.doc/InfoCenterTour800600.htm>.

### **Searching product support documents**

If you need to look beyond the information center to answer your question or resolve your problem, you can use one or more of the following approaches:

- Find the content that you need by using the IBM Support Portal at [www.ibm.com/software/support](http://www.ibm.com/software/support) or directly at [www.ibm.com/support/entry/portal](http://www.ibm.com/support/entry/portal).

The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services. The IBM

Support Portal lets you access the IBM electronic support portfolio from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution.

Familiarize yourself with the IBM Support Portal by viewing the demo videos at [https://www.ibm.com/blogs/SPNA/entry/the\\_ibm\\_support\\_portal\\_videos?lang=en\\_us](https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos?lang=en_us) about this tool. These videos introduce you to the IBM Support Portal, explore troubleshooting and other resources, and demonstrate how you can tailor the page by moving, adding, and deleting portlets.

Access a specific IBM Software Support site:

- Application Performance Analyzer for z/OS Support
- Debug Tool for z/OS Support
- Enterprise COBOL for z/OS Support
- Enterprise PL/I for z/OS Support
- Fault Analyzer for z/OS Support
- File Export for z/OS Support
- File Manager for z/OS Support
- WebSphere Developer Debugger for System z Support
- WebSphere Studio Asset Analyzer for Multiplatforms Support
- Workload Simulator for z/OS and OS/390 Support
- Search for content by using the IBM masthead search. You can use the IBM masthead search by typing your search string into the Search field at the top of any ibm.com page.
- Search for content by using any external search engine, such as Google, Yahoo, or Bing. If you use an external search engine, your results are more likely to include information that is outside the ibm.com domain. However, sometimes you can find useful problem-solving information about IBM products in newsgroups, forums, and blogs that are not on ibm.com. Include "IBM" and the name of the product in your search if you are looking for information about an IBM product.
- The IBM Support Assistant (also referred to as ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. It provides quick access to support-related information. You can use the IBM Support Assistant to help you in the following ways:
  - Search through IBM and non-IBM knowledge and information sources across multiple IBM products to answer a question or solve a problem.
  - Find more information through product and support pages, customer news groups and forums, skills and training resources and information about troubleshooting and commonly asked questions.

In addition, you can use the built-in Updater facility in IBM Support Assistant to obtain IBM Support Assistant upgrades and new features to add support for more software products and capabilities as they become available.

For more information, and to download and start using the IBM Support Assistant for IBM System z Enterprise Development Tools & Compilers products, please visit [http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&dc=D600&uid=swg21242707&loc=en\\_US&cs=UTF-8&lang=en](http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&dc=D600&uid=swg21242707&loc=en_US&cs=UTF-8&lang=en).

General information about the IBM Support Assistant can be found on the IBM Support Assistant home page at <http://www.ibm.com/software/support/isa>.

---

## Getting fixes

A product fix might be available to resolve your problem. To determine what fixes and other updates are available, select a link from the following list:

- Latest PTFs for Application Performance Analyzer for z/OS
- Latest PTFs for Debug Tool for z/OS
- Latest PTFs for Fault Analyzer for z/OS
- Latest PTFs for File Export for z/OS
- Latest PTFs for File Manager for z/OS
- Latest PTFs for Optim Move for DB2
- Latest PTFs for WebSphere Studio Asset Analyzer for Multiplatforms
- Latest PTFs for Workload Simulator for z/OS and OS/390

When you find a fix that you are interested in, click the name of the fix to read its description and to optionally download the fix.

Subscribe to receive email notifications about fixes and other IBM Support information as described in [Subscribing to Support updates..](#)

---

## Subscribing to support updates

To stay informed of important information about the IBM products that you use, you can subscribe to updates. By subscribing to receive updates, you can receive important technical information and updates for specific Support tools and resources. You can subscribe to updates by using the following:

- RSS feeds and social media subscriptions
- My Notifications

### RSS feeds and social media subscriptions

For general information about RSS, including steps for getting started and a list of RSS-enabled IBM web pages, visit the IBM Software Support RSS feeds site at <http://www.ibm.com/software/support/rss/other/index.html>. For information about the RSS feed for the IBM System z Enterprise Development Tools & Compilers information center, refer to the [Subscribe to information center updates](#) topic in the information center at [http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/topic/com.ibm.help.doc/subscribe\\_info.html](http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/topic/com.ibm.help.doc/subscribe_info.html).

### My Notifications

With My Notifications, you can subscribe to Support updates for any IBM product. You can specify that you want to receive daily or weekly email announcements. You can specify what type of information you want to receive (such as publications, hints and tips, product flashes (also known as alerts), downloads, and drivers). My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

To subscribe to Support updates, follow the steps below.

1. Click My notifications to get started. Click **Subscribe now!** on the page.
2. Sign in My notifications with your IBM ID. If you do not have an IBM ID, create one ID by following the instructions.

3. After you sign in My notifications, enter the name of the product that you want to subscribe in the **Product lookup** field. The look-ahead feature lists products matching what you typed. If the product does not appear, use the **Browse for a product** link.
4. Next to the product, click the **Subscribe** link. A green check mark is shown to indicate the subscription is created. The subscription is listed under Product subscriptions.
5. To indicate the type of notices for which you want to receive notifications, click the **Edit** link. To save your changes, click the **Submit** at the bottom of the page.
6. To indicate the frequency and format of the email message you receive, click **Delivery preferences**. Then, click **Submit**.
7. Optionally, you can click the RSS/Atom feed by clicking **Links**. Then, copy and paste the link into your feeder.
8. To see any notifications that were sent to you, click **View**.

---

## Contacting IBM Support

IBM Support provides assistance with product defects, answering FAQs, and performing rediscovery.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company must have an active IBM maintenance contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the information below or refer to the Support portfolio topic in the Software Support Handbook at <http://www14.software.ibm.com/webapp/set2/sas/f/handbook/offering.html>.

- For IBM distributed software products (including, but not limited to, Tivoli®, Lotus®, and Rational® products, as well as DB2 and WebSphere® products that run on Windows, or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:

### Online

Go to the Passport Advantage Web site at [http://www.lotus.com/services/passport.nsf/WebDocs/Passport\\_Advantage\\_Home](http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home) and click **How to Enroll**.

### By phone

For the phone number to call in your country, go to the Contacts page of the *IBM Software Support Handbook* on the web at <http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html> and click the name of your geographic region.

- For customers with Subscription and Support (S & S) contracts, go to the Software Service Request Web site at <http://www.ibm.com/support/servicerequest>.
- For customers with IBMLink, CATIA, Linux, S/390®, iSeries, pSeries, zSeries, and other support agreements, go to the IBM Support Line Web site at <http://www.ibm.com/services/us/index.wss/so/its/a1000030/dt006>.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web site at <http://www.ibm.com/servers/eserver.techsupport.html>.

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States. From other countries, go to the Contacts page of the *IBM Software Support Handbook* on the web at <http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html> and click the name of your geographic region for phone numbers of people who provide support for your location.

Complete the following steps to contact IBM Support with a problem:

1. “Define the problem and determine the severity of the problem”
2. “Gather diagnostic information”
3. “Submit the problem to IBM Support” on page 566

To contact IBM Software support, follow these steps:

## Define the problem and determine the severity of the problem

Define the problem and determine severity of the problem. When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Support can help you solve the problem efficiently.

IBM Support needs you to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting. Use the following criteria:

### Severity 1

The problem has a **critical** business impact. You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.

### Severity 2

The problem has a **significant** business impact. The program is usable, but it is severely limited.

### Severity 3

The problem has **some** business impact. The program is usable, but less significant features (not critical to operations) are unavailable.

### Severity 4

The problem has **minimal** business impact. The problem causes little impact on operations, or a reasonable circumvention to the problem was implemented.

For more information, see the Getting IBM support topic in the Software Support Handbook at <http://www14.software.ibm.com/webapp/set2/sas/f/handbook/getsupport.html>.

## Gather diagnostic information

To save time, if there is a Mustgather document available for the product, refer to the Mustgather document and gather the information specified. Mustgather documents contain specific instructions for submitting your problem to IBM and gathering information needed by the IBM support team to resolve your problem. To determine if there is a Mustgather document for this product, go to the product support page and search on the term Mustgather. At the time of this publication, the following Mustgather documents are available:

- Mustgather: Read first for problems that are encountered with Application Performance Analyzer for z/OS: [http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&q1=mustgather&uid=swg21265542&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&q1=mustgather&uid=swg21265542&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems that are encountered with Debug Tool for z/OS: [http://www.ibm.com/support/docview.wss?rs=615&context=SSGTS&q1=mustgather&uid=swg21254711&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=615&context=SSGTS&q1=mustgather&uid=swg21254711&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems that are encountered with Fault Analyzer for z/OS: [http://www.ibm.com/support/docview.wss?rs=273&context=SSXJAJ&q1=mustgather&uid=swg21255056&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=273&context=SSXJAJ&q1=mustgather&uid=swg21255056&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems that are encountered with File Manager for z/OS: [http://www.ibm.com/support/docview.wss?rs=274&context=SSXJAV&q1=mustgather&uid=swg21255514&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=274&context=SSXJAV&q1=mustgather&uid=swg21255514&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems that are encountered with Enterprise COBOL for z/OS: [http://www.ibm.com/support/docview.wss?rs=2231&context=SS6SG3&q1=mustgather&uid=swg21249990&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=2231&context=SS6SG3&q1=mustgather&uid=swg21249990&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems that are encountered with Enterprise PL/I for z/OS: [http://www.ibm.com/support/docview.wss?rs=619&context=SSY2V3&q1=mustgather&uid=swg21260496&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=619&context=SSY2V3&q1=mustgather&uid=swg21260496&loc=en_US&cs=utf-8&lang=en)

If the product does not have a Mustgather document, please provide answers to the following questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can you recreate the problem? If so, what steps were performed to recreate the problem?
- Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, and so on.
- Are you currently using a workaround for the problem? If so, be prepared to explain the workaround when you report the problem.

## Submit the problem to IBM Support

You can submit your problem to IBM Support in one of three ways:

### Online using the IBM Support Portal

Click **Service request** on the IBM Software Support site at <http://www.ibm.com/software/support>. On the right side of the Service request page, expand the Product related links section. Click Software support (general) and select ServiceLink/IBMLink to open an Electronic Technical Response (ETR). Enter your information into the appropriate problem submission form.

### Online using the Service Request tool

The Service Request tool can be found at <http://www.ibm.com/software/support/servicerequest>.

### By phone

Call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the Contacts page of the *IBM Software Support Handbook* at <http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html> and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

After a Problem Management Record (PMR) is open, you can submit diagnostic MustGather data to IBM using one of the following methods:

- FTP diagnostic data to IBM. For more information, refer to <http://www.ibm.com/support/docview.wss?rs=615&uid=swg21154524>.
- If FTP is not possible, email diagnostic data to [techsupport@mainz.ibm.com](mailto:techsupport@mainz.ibm.com). You must add PMR xxxxx bbb ccc in the subject line of your email. xxxxx is your PMR number, bbb is your branch office, and ccc is your IBM country code. Go to <http://itcenter.mainz.de.ibm.com/ecurep/mail/subject.html> for more details.

Always update your PMR to indicate that data was sent. You can update your PMR online or by phone as described above.



---

## Appendix B. Sample customized ISPF interface front-end

In certain circumstances it might be desirable to dynamically tailor the initial Fault Entry List display shown when the Fault Analyzer ISPF interface is invoked. For example, to pre-select the Fault History File or View name being used, or provide a dynamically created MATCH command. A MATCH command can be useful, for example to MATCH on today's date, or a specific PROGRAM name.

An example of how this dynamic tailoring can be achieved is included in the samples data set (IDI.SIDISAM1). The example displays a popup panel which allows the user to supply an optional program name and an Application ID or 'View' name (see Figure 179 on page 570). A '?' can be placed in the Application/View field to display a list of available applications and views (see Figure 180 on page 570). In the sample, if the length of the Application/View ID is 2, then the selected name is used to form the name of a Fault History File as follows:

```
<Variable DSNp1>.<System ID>.HIST.<Variable DSNp2>.<Application>
```

If not a length of 2, then the ID is assumed to be the name of a Fault Analyzer VIEW.

Once an Application/View ID has been successfully entered, and its existence verified, then the user can press Enter to invoke Fault Analyzer. If a program name was also supplied, then a corresponding MATCH command is also created.

The sample comprises the following files, each of which corresponds to a member name in the IDI.SIDISAM1 data set. Each file should be copied to a data set that is concatenated to the DDname indicated in the table.

*Table 25. Sample files*

File	DDname	Description
IDISFEMA	SYSPROC	Main REXX exec
IDISFESK	ISPSSLIB	ISPF skeleton for creating history file
IDISFECL	SYSPROC	Intermediary CLIST used when invoking Fault Analyzer
IDISFEAP	ISPPLIB	ISPF panel used for application selection
IDISFEQP	ISPPLIB	Query ISPF panel
IDISFEMP	ISPPLIB	Main ISPF panel for supplying user parameters

## Sample customized ISPF interface front-end

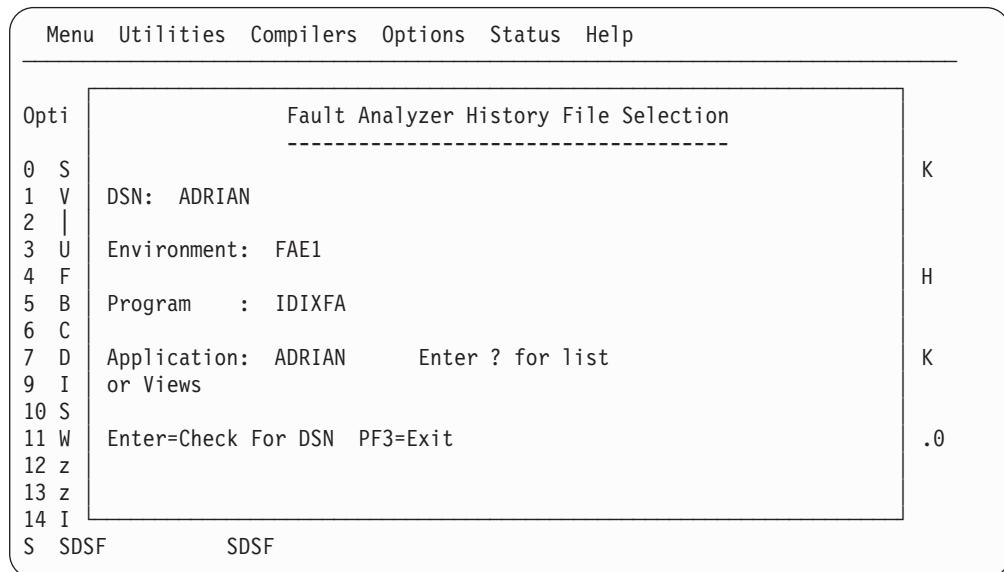


Figure 179. Sample display 1

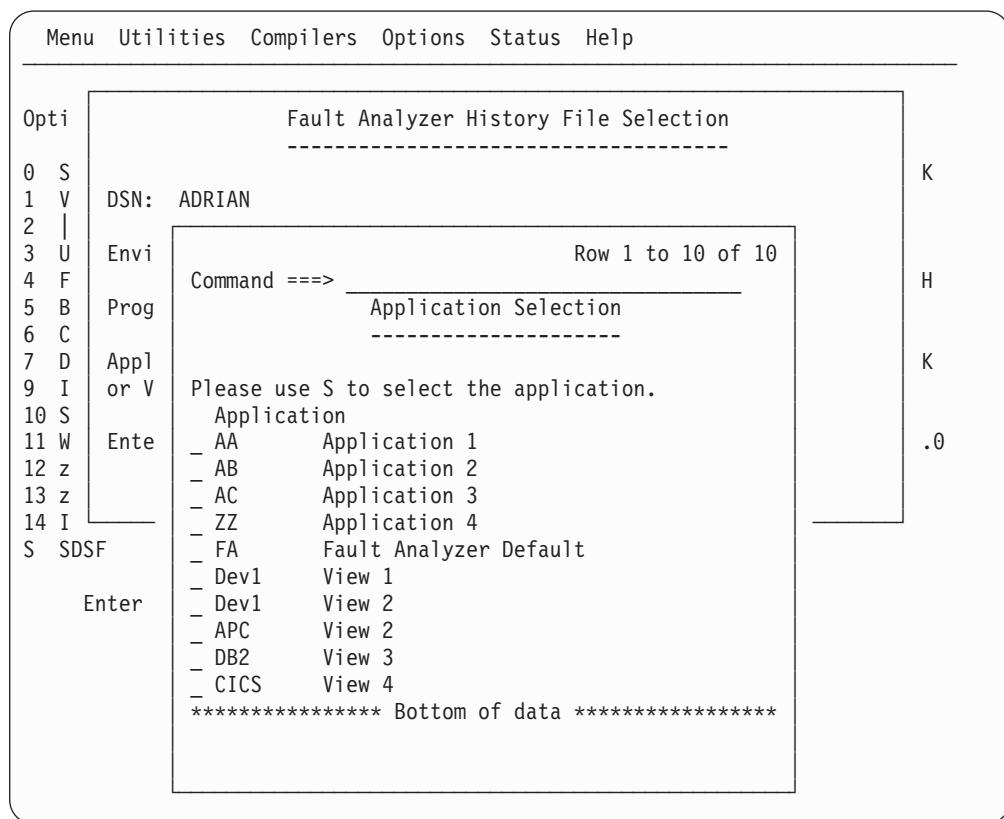


Figure 180. Sample display 2

Once a Fault History File or View name has been selected and verified, the sample code performs the following processing.

---

## Fault History File or VIEW name

In the Fault Analyzer ISPF application (IDI) variable pool, there is a variable called OLDHIST, which is a list of the last 10 accessed Fault History Files or Views. The first item in this list is the History File or View which is opened by the Fault Analyzer ISPF interface. The sample code modifies this list, such that the first item in the list is the History File or View entered by the user. It does this modification by scanning the list to see if the name already exists, in which case the entry is moved to the top of the list. If the name does not exist, then it is inserted at the top of the list, and all other entries are moved down one position, that is item 10 (if it exists) disappears.

In the list, View names are distinguished from Fault History Files by being enclosed within parentheses.

---

## MATCH on program name

If a program name has been entered, then the following command string is created:

```
MATCH PROGRAM <supplied program name>
```

If a program name has not been supplied, then a MATCH ALL command is created.

This command string is picked up when the Fault Analyzer ISPF interface starts, and is executed accordingly.

---

## Installing the sample application

It is important that the sample application (IDISFEMA) executes using the same ISPF application ID as the main Fault Analyzer ISPF application. If it does not, then updates made by the sample to ISPF variables are not correctly picked up. A way to use the sample is to add a new ISPF command, which invokes the main sample program using the same ISPF NEWAPPL application as is used for the 'normal' invocation of Fault Analyzer. For example, add the following command to a 'USER' command table, which is used by Fault Analyzer ISPF users:

Verb	T	Action
FASEL	0	SELECT CMD(%IDISFEMA &ZPARM) NEWAPPL(IDI)

This command assumes that Fault Analyzer is normally invoked using a NEWAPPL of IDI, for example from an application selection panel using a command similar to the following:

```
9,PGM(IDIPDDIR) NEWAPPL(IDI) SCRNAME(FAULTA)
```

As well as the above, the CLIST IDISFECL must be in one of the data sets that is allocated to the user's SYSPROC concatenation.

---

## How the sample works

So that the MATCH command string can be passed and executed by the Fault Analyzer ISPF interface, an intermediary CLIST is used (IDISFECL). This CLIST is invoked by passing an invocation command in the COMMAND option of an ISPF DISPLAY PANEL command. When the COMMAND option is specified on a DISPLAY PANEL command, then the actual PANEL referenced is not displayed,

## How the sample works

and the command in the COMMAND options is executed. For example, if a program name of MYPROG1 had been entered, then the following command string is created:

```
TSO IDISFECL;;MATCH PROGRAM MYPROG1
```

This command string, which is assigned to variable CMDSTACK in the sample code, is referenced in the ISPF DISPLAY command as follows:

```
Address ISPEXEC 'DISPLAY PANEL(IDISFEAP) COMMAND(CMDSTACK)'
```

---

## Appendix C. Java API to download Fault Analyzer report

This API downloads a Fault Analyzer report from the host and shows it in Eclipse-based products that are developed in Java. The download part of the process happens only if the Fault Analyzer report is not already downloaded. This API belongs to the FAAPI package, which is available as part of the Fault Analyzer plug-in for Eclipse products. The assumption is that the specified host ID and port number is defined in the Systems Information view.

---

### Specification

Two interfaces are designed to achieve this function:

```
FAAPI.openReport(String UniqueAddressOfFaultEntry);
    //UniqueAddressOfFaultEntry :"hostId/portNo/historyFile/faultId"
FAAPI.openReport(String hostId, int portNo, String historyFile, String faultId);
```

Both interfaces perform the same function, only the style of argument-specification differs.

All arguments are case-sensitive.

Here is an example that uses both interfaces to open fault entry F03004 in history file DA.DCAT on system pthfae1, which listens on port 7799:

```
FAAPI.openReport("pthfae1/7799/DA.DCAT/F03004");
FAAPI.openReport("pthfae1", 7799, "DA.DCAT", "F03004");
```

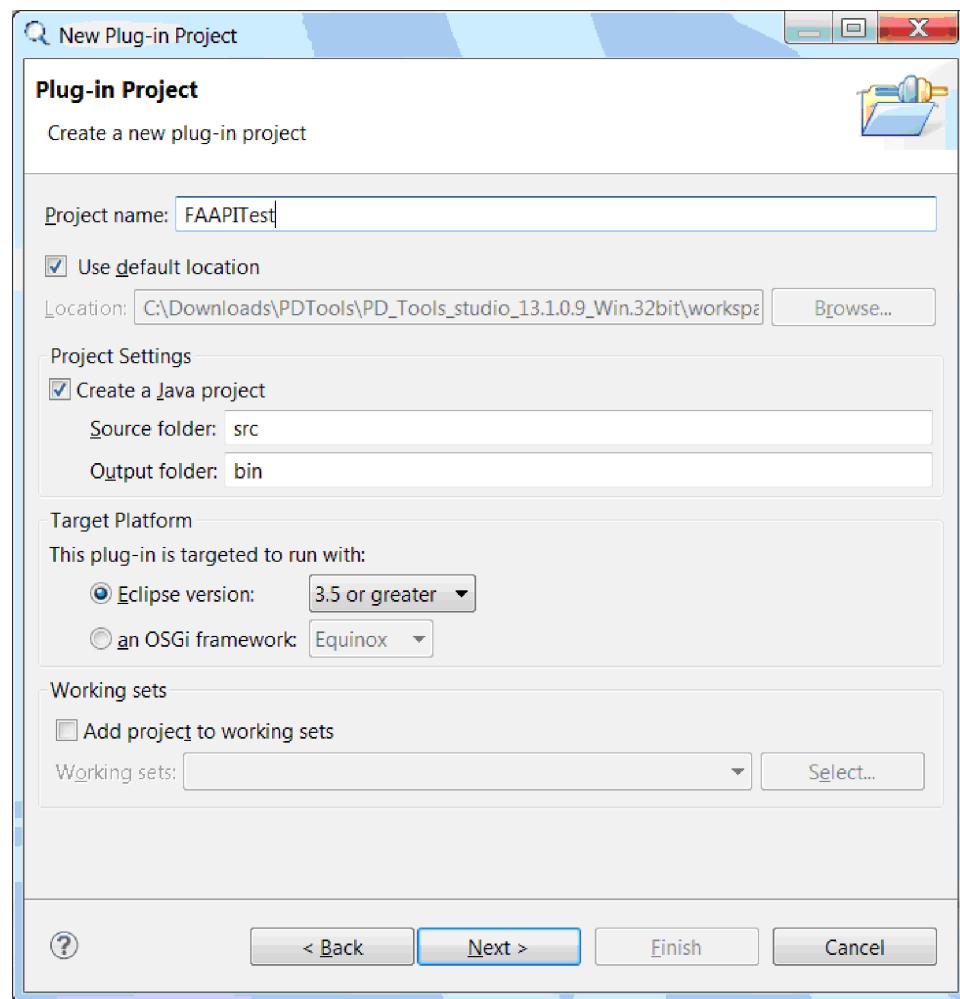
---

### Example

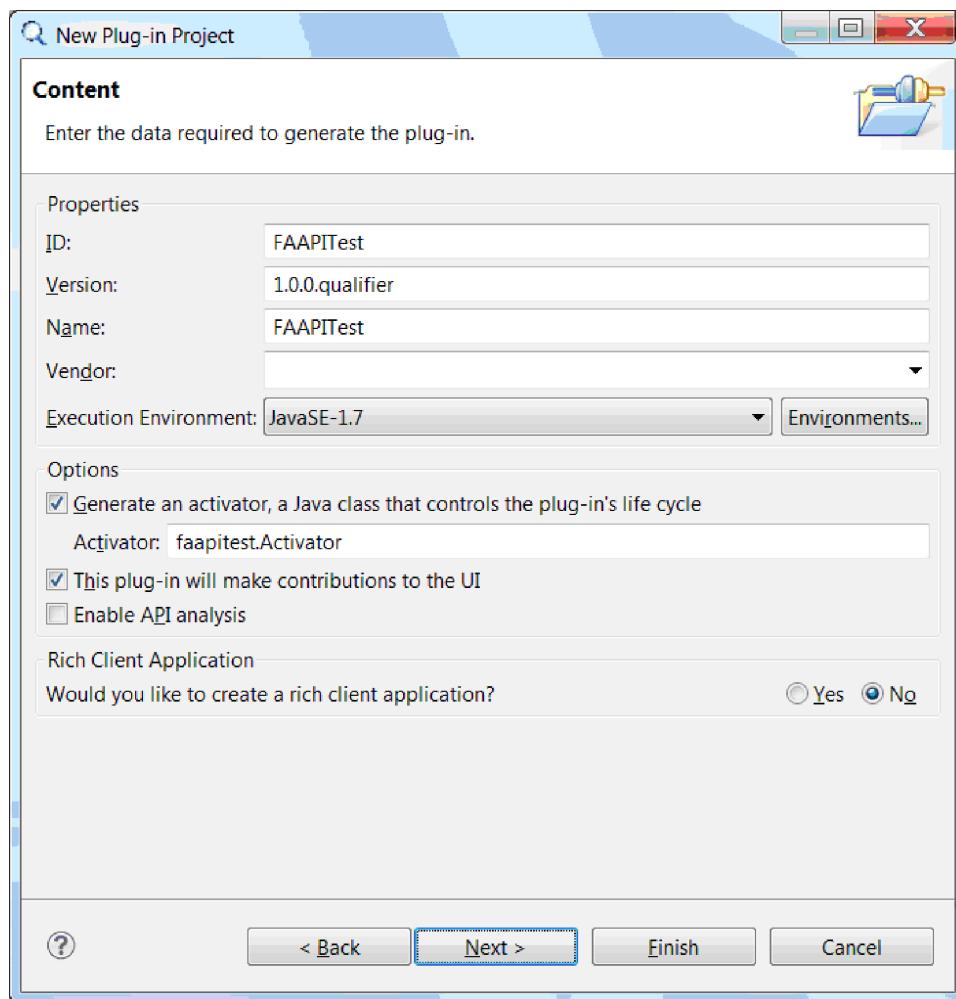
This example shows how to extend the HelloWorld sample plug-in in Eclipse to open a Fault Analyzer report. Follow these steps to create the HelloWorld sample plug-in, and then modify it to download and open a Fault Analyzer report.

1. Activate Java perspective in PD Tools Studio and click **New -> Plug-in Project**.

## Java API to download Fault Analyzer report

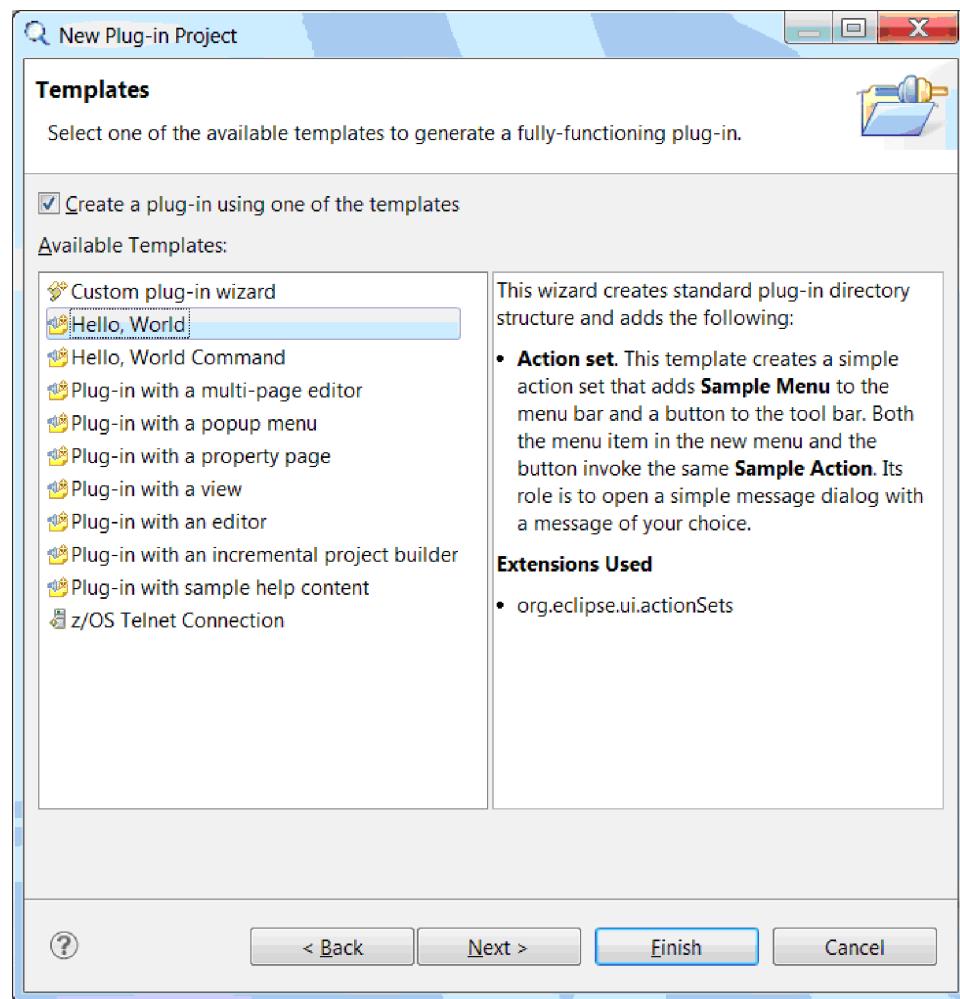


2. Enter a Project name (in this sample FAAPITest) and click **Next**.

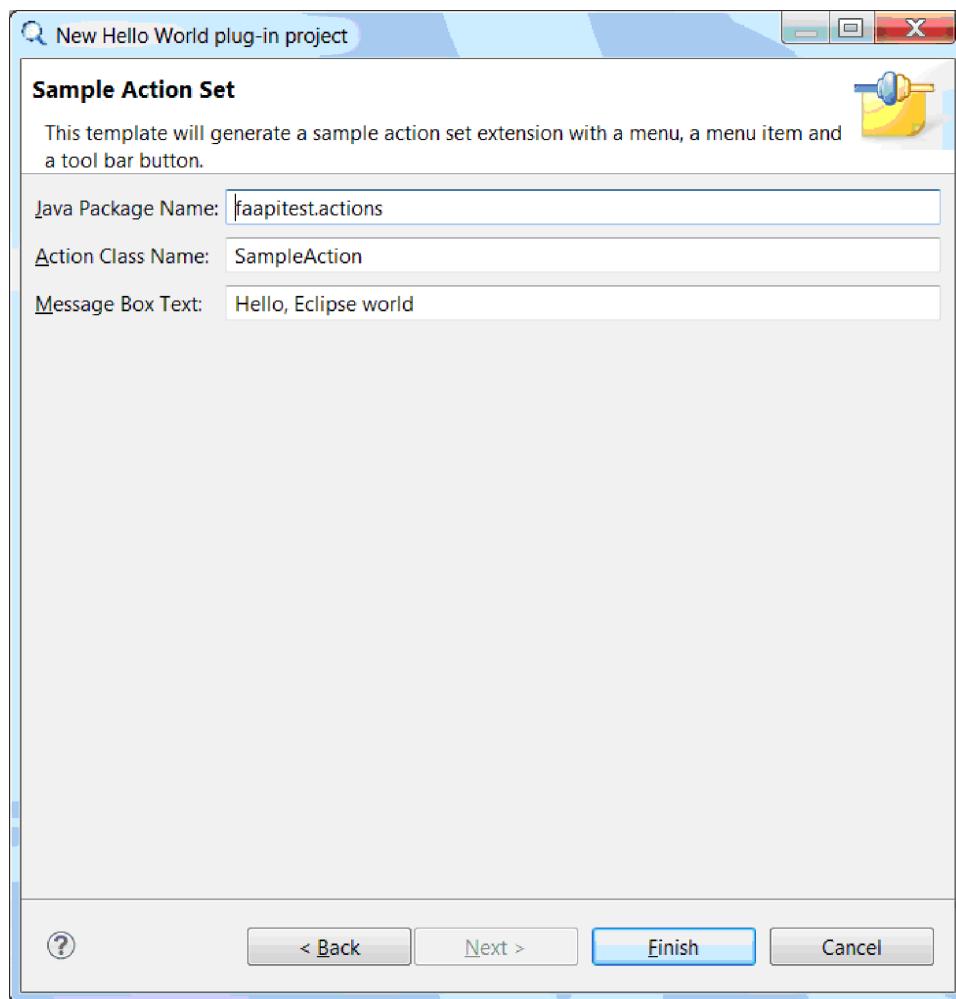


3. Click **Next**.

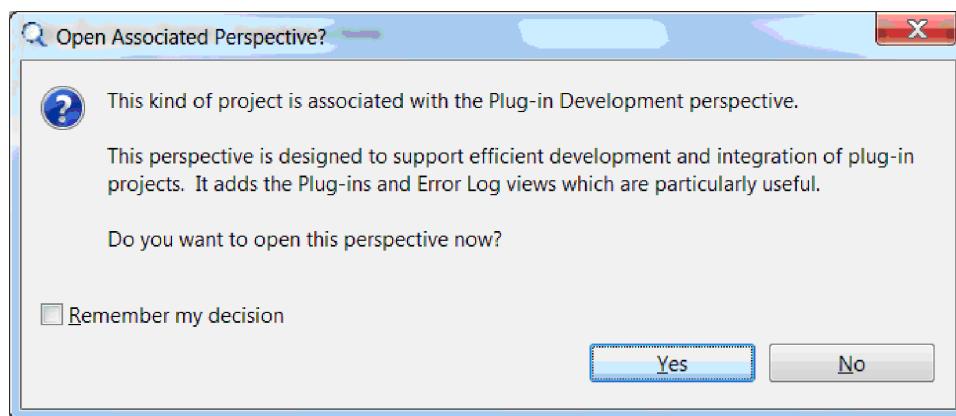
## Java API to download Fault Analyzer report



4. Select the **Hello, World** template and click **Next**.



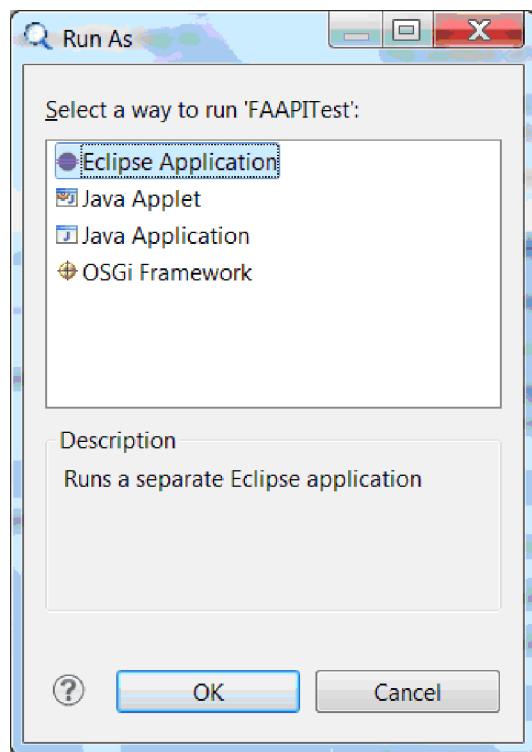
5. Click **Finish**. If you receive the following message, click **Yes**.



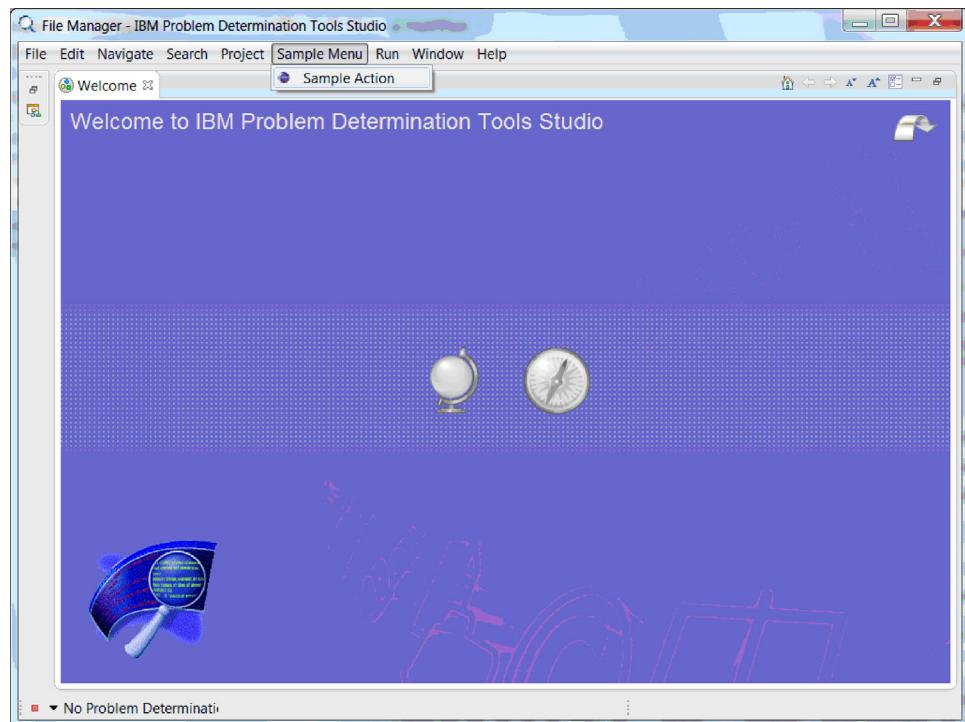
At this stage, FAAPITest is created. To run the generated plug-in at this stage, follow these steps:

1. Click **FAAPITest** in Package Explorer.
2. Click **Run** from the main menu and choose Run.
3. Select **Eclipse Application** and click **OK**.

## Java API to download Fault Analyzer report



4. The running environment, displays the following window. Click **Sample Menu** and select **Sample Action**.



5. The following message appears on the screen:



6. Click **OK** and close the running Eclipse environment.

Now change the FAAPITest plug-in to show a fault entry when you click **Sample Menu -> Sample Action**.

Here are the steps:

1. Open MANIFEST.MF. This file exists in the META-INF folder of the FAAPITest plug-in.
2. Click the **Dependencies** tab. Then click **Add...** for required plug-ins and select **com.ibm.etools.fa.pdtclient.ui**. Click **OK** and save the MANIFEST file.
3. Open **faapitest.actions.SampleAction.java** and modify the run method as shown in the following example:

```
....  
public void run(IAction action) {  
    //CHANGE THE PARAMETERS BASED ON YOUR SETUP  
    FAAPI.openReport("pthfae1", 7799, "DA.DCAT", "F03004");  
    //MessageDialog.openInformation(  
    //    window.getShell(),  
    //    "FAAPITest",  
    //    "Hello, Eclipse world");  
}  
....
```

## Java API to download Fault Analyzer report



```
FAAPITest SampleAction.java
package faapitest.actions;
import org.eclipse.jface.action.IAction;
/*
 * Our sample action implements workbench action delegate.
 * The action proxy will be created by the workbench and
 * shown in the UI. When the user tries to use the action,
 * this delegate will be created and execution will be
 * delegated to it.
 * @see IWorkbenchWindowActionDelegate
 */
public class SampleAction implements IWorkbenchWindowActionDelegate {
    private IWorkbenchWindow window;
    /**
     * The constructor.
     */
    public SampleAction() {
    }
    /**
     * The action has been activated. The argument of the
     * method represents the 'real' action sitting
     * in the workbench UI.
     * @see IWorkbenchWindowActionDelegate#run
     */
    public void run(IAction action) {
        FAAPI.openReport("pthfae1", 7799, "DA.DCAT", "F03004");
        //MessageDialog.openInformation(
        //    window.getShell(),
        //    "FAAPITest",
        //    "Hello, Eclipse world");
    }
}
```

4. Run the modified plug-in as explained earlier.
5. Make sure that the system is defined in the Systems Information view. If not, add the system.
6. Click **Sample Menu -> Sample Application**. The report is downloaded and displayed in the current perspective in the Eclipse editor area.

---

## **Appendix D. Technical details for screen size adjustments**

To support the widest range of terminal characteristics, use a DLOGMOD specification of D4C32XX3, in the IBM supplied MODETAB of ISTINCLM. Further information about this and Telnet server requirements can be found in z/OS Communications Server publications.

Users of IBM Personal Communications should reference IC71220: PCOM: HOW CAN YOU USE A 62X160 SCREEN SIZE? on the IBM Support Portal.



---

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

- IBM Director of Licensing
- IBM Corporation
- North Castle Drive
- Armonk, NY 10504-1785
- U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this at any time without notice.

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

IBM Corporation  
J46A/G4

## Notices

555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

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

---

## Programming interface information

This User's Guide and Reference documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Fault Analyzer.

---

**Trademarks**

A current list of IBM trademarks is available on the web at "Copyright and trademark information", <http://www.ibm.com/legal/copytrade.shtml>.



---

## Glossary

This glossary defines terms and abbreviations that are used in this book. If you do not find the term you are looking for refer to the index, or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

### A

**abend** Abnormal end of task; the termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

### analysis

The methodical investigation of a problem, and the separation of the problem into smaller related units for further study.

### C

**cache** A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

### compiler listing

A printout produced by compiling a program or creating a file and that optionally includes, for example, a line-by-line source listing, cross-reference list, diagnostic information, and for programs, a description of externally described files.

### D

**dump** To copy the contents of all or part of virtual storage to collect error information.

### E

### expert system

A system that provides for solving problems in a particular application area by drawing inferences from a knowledge base acquired by human expertise.

### F

### fault entry

The information about a fault saved as a member in a *history file* at the end of *real-time analysis*. The fault entry might also include a *minidump*.

### history file

A PDSE data set containing *fault entries* as individual members.

### L

### LANGX side file

A compiler listing or SYSADATA file, converted to a binary format using the Problem Determination Tools Common Component IPVLANGX utility, or one of its aliases CAZLANGX, EQALANGX or IDILANGX.

**listing** See *compiler listing*.

### M

### minidump

The storage pages that were referenced during real-time analysis by Fault Analyzer and saved in the history file entry for the fault. A minidump permits later reanalysis of the fault even if no SYSMDUMP was written.

### O

**option** A parameter that provides control on the operation of Fault Analyzer.

### P

### partitioned data set (PDS)

A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

### partitioned data set extended (PDSE)

A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

### PDS(E)

A data set that can be either a *partitioned data set (PDS)* or a *partitioned data set extended (PDSE)*.

### PDTCC

IBM Problem Determination Tools for z/OS Common Component.

### R

**real-time analysis**

The *analysis* undertaken by Fault Analyzer immediately after a program has *abended*.

**reanalysis**

A second or subsequent *analysis*.

**Recovery fault recording (RFR)**

A feature of Fault Analyzer which enables abnormal termination of a Fault Analyzer real-time analysis to still create a fault entry of the original application abend.

**RFR dump**

A SDUMP or IEATDUMP data set written during recovery fault recording processing, and associated with an RFR fault entry.

**RFR fault entry**

A history file fault entry written during recovery fault recording processing, and associated with an RFR dump data set.

**S****saved report**

The report that is contained within the fault entry and which can be viewed from the ISPF interface without the need to perform reanalysis.

**side file**

A condensed version of a *listing*, readable by computers (but not humans).

**system dump**

See *dump*.

**T****TDUMP**

Synonymous with IEATDUMP—a dump data set type used by Fault Analyzer recovery fault recording processing.

**V**

**view** A member of a *PDS(E)* containing a collection of fault history file data set names that are to be displayed simultaneously using the Fault Analyzer ISPF interface.

---

# Index

## Special characters

\_IDI\_OFF environment variable 357  
\_IDI\_OPTS environment variable  
  options 455  
\_IDI\_OPTSFILE environment variable  
  options 454  
-DROPCNF- data set name 460  
-DROPCNF- user exit name 475  
-HistCols 264  
-Match 264  
\$\$BACKUP history file data set  
  member 9  
\$\$INDEX data  
  caching 240  
\$\$INDEX history file data set member 9  
\$\$UFMTX example 165  
&SYSCLONE MVS system symbol  
  IDICNFxx parmlib member  
    suffix 279  
++IDIOPT1 256  
++IDIOPT2 256

## Numerics

64-bit addresses  
  specifying 93

## A

ABCODE  
  EXEC CICS ABEND command 305  
ABCODE keyword on EXEC CICS  
  ABEND command 305  
abend code explanation repository  
  setting up 233  
abend code explanations  
  displaying 77  
  user-defined 350  
abend codes  
  expanding 144  
abend job information 139, 181  
abend job information display  
  example 139, 182  
about Fault Analyzer  
  help menu option 62  
about Fault Analyzer display  
  example 80  
action-bar pull-down menus 61  
ADATA option  
  producing SYSADATA file 285  
add blank lines  
  view menu option 63  
add help text  
  view menu option 63  
add pseudo-assembler instructions  
  view menu option 63  
AdditionalIDIOffDD option 455  
ADDR tag 428  
address space  
  selecting for analysis 179

address TSO REXX commands 362  
allocate ISPF data sets 247  
analysis  
  real time  
    *See real-time analysis*  
Analysis Control user exit 368  
Analysis Control user exit (MVS SVC  
  dump registration) 371  
analysis engine 3  
analysis process 3  
analysis report  
  *See also reanalyzing*  
  real-time 18  
analyze MVS dump data set  
  file menu option 61  
analyze MVS dump data set display  
  example 178  
Analyzer  
  disabling 355  
application error handling  
  effect on invocation of Fault  
    Analyzer 229  
application-handled error conditions 229  
application-specific descriptions  
  providing 349  
application-specific explanations  
  providing 349  
AREA tag 428  
assembler  
  verifying Fault Analyzer 325  
assembler ADATA option 285  
assembler exits  
  CICS NoDup(CICSFAST)  
    override 310  
assembler invocation  
  IDISNAP 24  
associated file control blocks display  
  example 119  
associated storage areas display  
  example 141  
associated storage areas display with hex  
  value column collapsed example 142  
associated storage areas display with  
  level 88 items collapsed example 143  
AUTO-managed PDSE history files 262  
auxiliary trace data sets  
  formatting 187  
available dump status 86

## B

batch dump reanalysis  
  submitting 52  
batch options line 96  
batch reanalysis 7  
  data sets 100  
  initiating 100  
  JCL control statements 250  
  options 95  
  purpose 95

batch reanalysis options  
  options menu option 63  
batch reanalysis options display  
  example 96, 98  
batch utility  
  IDIUTIL 337  
binder-related dependencies 11  
blank lines  
  adding 76  
  removing 76

**C**

C  
  verifying Fault Analyzer 328  
CA-Panexec  
  exit for 256  
call depth  
  maximum 202  
CE command 64  
CEEWUCHA  
  special processing 13  
CEEWUCHA LE user condition  
  handler 197  
CFA 305  
  FORCEPURGE currently analyzed  
    task 303  
CFA Exit Options display example 307  
CFA IVP testing display example 330  
CFA transaction display example 306  
change fault history file settings  
  file menu option 62  
change fault history file settings display  
  example 57  
checklist  
  customizing Fault Analyzer 217  
  installing Fault Analyzer 217  
CICS  
  controlling transaction abend  
    analysis 305  
  criteria for duplicate fault  
    determination 488  
  customization  
    verifying 329  
  defining required programs 302  
  ensuring transaction abend analysis  
    not suppressed by DUMP(NO) 310  
IDIOPTS DDname 312  
IVP testing 329  
performing interactive reanalysis 206  
storage requirements 313  
  verifying customization 329  
CICS auxiliary trace data set formatting  
  selecting trace data set 187  
CICS auxiliary trace data sets  
  formatting 187  
CICS channels  
  system-wide information 126  
CICS commareas  
  system-wide information 126

CICS definitions  
     sample job 304

CICS dump data set  
     CICS transaction LE dump  
         output 312

CICS environment  
     customizing File Manager 301

CICS Explorer plug-in  
     customizing IBM Problem  
         Determination Tools for z/OS  
         Common Component Common  
         Server 443

downloading CICS Explorer 444

downloading Fault Analyzer  
     plug-in 444

install Fault Analyzer plug-in into  
     CICS Explorer 444

installing CICS Explorer 444

CICS fast duplicate fault  
     suppression 394, 489

CICS global user exits 16

CICS information  
     system-wide 119

CICS information display example 119

CICS levels  
     system-wide information 126

CICS levels, commareas, and channels  
     display example 126

CICS NoDup(CICSFAST) override  
     assembler exit (IDINDFUE) 310

CICS open (L9) TCB  
     capturing abends on 313

CICS region SDSF job data set display  
     example 308

CICS shutdown PLT  
     *See* shutdown PLT

CICS startup  
     installing Fault Analyzer 303

CICS Storage Accounting Area (SAA)  
     overlays  
         assistance 13

CICS system abend analysis  
     creating history file entry 185  
     displaying interactive report 180  
     selecting dump data set 177  
     setting options 177  
     user exit usage 177

CICS system abend dump analysis 177

CICS system abend interactive reanalysis  
     report display example 180

CICS system abend synopsis display  
     example 181

CICS system information 182

CICS system information display  
     example 183

CICS trace  
     preventing wrap from LE 312

CICS trace considerations 312

CICS trace display example 126

CICS trace formatting  
     system-wide information 123

CICS Trace Selection Parameters  
     specifying 188

CICS trace selection parameters display  
     example 124

CICS trace, formatting  
     system-wide information 123

CICS trace, summarized  
     system-wide information 122

CICS tracing 305

CICS transaction abend analysis  
     maximizing performance 314

CICS transaction abends  
     enabling dynamic control of  
         analysis 303  
     special handling 258

CICS transaction LE dump output  
     directing to CICS dump data set 312

CICS transaction storage summary  
     display example 120

CICS XDUREQ global user exit  
     stopping Fault Analyzer analysis 258

CICSDumpTableExclude option 456

clear last accessed information  
     file menu option 62

COBOL  
     verifying Fault Analyzer 326

COBOL base locators  
     showing 144

COBOL Explorer 172  
     example 173

COBOL invocation  
     IDISNAP 23

COBOL Report Writer Precompiler 286

COBOL suppressed copybooks 199

COBOL SYSDEBUG file usage 288, 290

COLS command 64

column  
     hex-value  
         *See* hex-value column

column configuration  
     view menu option 63

column configuration source  
     identification 44

column layout  
     specifying default 264

columns available for fault entry list  
     display 45

com.ibm.faultanalyzer.Snap.Dump  
     method  
         invoking Fault Analyzer from  
             Java 25

commands  
     CE 64  
     COLS 64  
     COPY 65  
     DISASM 65  
     DSECT 66  
     DUPS 66  
     EXEC 66  
     FIND 67  
     INFO 69  
     ISPF interface 64  
     LOOKUP 69  
     MATCH 70  
     NEXT 70  
     NOTELIST 71  
     PREV 71  
     QUIT 71  
     reading syntax diagrams xi  
     REFRESH 72  
     RESET 72  
     RPTFIND 72  
     RUNCHAIN 73

commands (*continued*)  
     SHOW 73  
     STCK 74  
     VIEWS 74

comments  
     in options 452

Compiler Listing display example 145,  
     146, 147

Compiler Listing Not Found display  
     example 167

Compiler Listing Read user exit 372

compiler listings  
     attributes 296  
     example of providing for  
         reanalysis 101  
     locating 290  
     naming 289  
     prompting display 166  
     providing 285  
     selection criteria 12  
     temporary data set 297

compiler options required for  
     IDILANGX 286

Confirm Exit display example 111

confirm fault entry deletion display  
     example 82

confirm fault entry deletion option  
     changing 81, 82  
     deleting when not set 82  
     deleting when set 81

Confirm Java Fault Entry Reanalysis  
     display example 191

confirm sysmdump open display  
     example 106

COPY command 65

copy current display to data set  
     services menu option 63

copyright  
     displaying 80

create history file entry display  
     example 186

Create Java Fault Entry display  
     example 190

criteria  
     compiler listing selection 12  
     side file selection 12

criteria for successful analysis 522

cross-system coupling facility 273

Cross-system Coupling Facility  
     (XCF) 240

CSECTs  
     naming 289

CTL data area 506

cultural environment  
     specifying 283

cursor match 51

customer support  
     *See* Software Support

customization  
     verifying 325

customization checklist 217

## D

data area decimal character field  
     getting data 438

data areas  
   displaying 149  
   displaying chained 157  
   field value 438  
 data set members  
   *See also* sample data set members  
   sample  
     *See* sample data set members  
 data set name substitution symbols 460  
 data set security server access  
   requirement  
     IDIDOC 234  
     IDIVSENU 234  
     IDIVSJPN 234  
     IDIVSKOR 234  
 data sets  
   batch reanalysis 100  
   dropping 460  
   interactive reanalysis 171  
   ISPF  
     *See* ISPF data sets  
     reanalysis 459  
 DATA tag 429  
 DataSets option 457  
 DB2  
   binding 317  
   IDIS subsystem requirements 243  
   LE considerations 317  
   verifying Analyzer 331  
 DB2 environment  
   customizing 317  
 DB2 information  
   system-wide information 129  
 DB2 information display example 129  
 DB2 IVP  
   C 331  
   COBOL 333  
 DB2 performance  
   improving 317  
 DD tag 429  
 DDnames  
   IDIRLOAD 297  
 DEBUG option considerations 288  
 default options  
   changing 279  
   installation-wide 279  
   setting 279  
 Deferred Breakpoints Feature 176  
 DeferredReport option 463  
 delete confirmation display  
   deleting when not shown 82  
   deleting when shown 81  
 DELETE control statement 339  
 deletion options  
   setting from Options menu 81  
 descriptor codes  
   WTO messages 13  
 Detail option 465  
 detailed event information 114  
 detailed PCB display 136  
 DFHRPL 305  
 DISASM command 65  
 display area  
   increasing 251  
 display information  
   setting preferred formatting width 76

displays  
   abend job information 139, 182  
   about Fault Analyzer 80  
   analyze MVS dump data set 178  
   associated file control blocks 119  
   associated storage areas 141  
   associated storage areas with hex  
     value column collapsed 142  
   associated storage areas with level 88  
   items collapsed 143  
   batch reanalysis options 96, 98  
   CFA IVP testing 330  
   CFA transaction 306, 307  
   change fault history file settings 57  
   CICS information 119  
   CICS levels, commareas, and  
     channels 126  
   CICS region SDSF job data set 308  
   CICS system abend interactive  
     reanalysis report 180  
   CICS system abend synopsis 181  
   CICS system information 183  
   CICS trace 126  
   CICS trace selection parameters 124  
   CICS Transaction Storage  
     Summary 120  
   compiler listing 146, 147  
   Compiler Listing 145  
   Compiler Listing Not Found 167  
   Confirm Exit 111  
   confirm fault entry deletion 82  
   Confirm Java Fault Entry  
     Reanalysis 191  
   confirm sysmdump open 106  
   create history file entry 186  
   Create Java Fault Entry 190  
   DB2 information 129  
   dump storage 147  
   event details 115  
   event summary 113  
   exclude program from side file  
     search 170  
   Fault Analyzer options 140  
   Fault Analyzer preferences 81  
   fault entry duplicate history 88  
   fault entry information 83  
   fault entry list 35, 36  
   fault entry list column  
     configuration 43  
   fault entry list with updates  
     pending 60  
   File Browse 42  
   file information 118  
   format CICS auxiliary trace data  
     set 187  
   Formatting User Exit Selection  
     List 165  
   hex-dumped storage 137  
   history file properties 53  
   history file updates pending 60  
   IMS information 131  
   IMS summary information 134  
   interactive reanalysis options 104  
   interactive reanalysis report 110  
   interactive reanalysis status 109  
   Java event details 194  
   Java event summary 193

displays (*continued*)  
   Java information 195  
   Java interactive reanalysis report 192  
   Language Environment heap analysis  
     information 138  
   last accessed history file entries 41  
   last accessed history files or views 40  
   last CICS 3270 screen buffer 121  
   last CICS 3270 screen buffer hex 122  
   level 88 items 143  
   listing/side file mismatch 169  
   Lookup Search and Browse 78  
   message explanation 144  
   message id look-up 78  
   MTRACE records 138  
   new history file allocation 56  
   options in effect 185  
   preferred formatting width 77  
   real-time report 75  
   specify compiler listing or side  
     file 168  
   specify move/copy options 90  
   specify XMIT options 92  
   STCK conversion 163  
   storage disassemble 161  
   storage DSECT mapping entry 155  
   storage DSECT mapping map 156  
   storage RUNCHAIN command  
     entry 158  
   summarized CICS trace 123  
   suppressed dump storage 149  
   synopsis 112  
   system-wide messages 129  
   system-wide open files 117  
   system-wide storage areas 136  
   user fields update prompt 87  
   user note list 153  
   user notes update prompt 154  
   view list 42  
 DL tag 429  
 DSECT command 66  
 DSECT data sets (\$DINDEX member)  
   indexing 157  
 DSECT indexing utility (IDIPDSCU) 157  
 DSECT information  
   mapping storage areas 154  
 DSNACLI default DB2 plan 317  
 DT tag 430  
 DUMMY data set  
   specifying via DataSets option 459  
 dump data set name  
   viewing 53  
 dump data set size 229  
 dump DD statement  
   eliminating need 255  
 dump registration  
   Analysis Control user exit 371  
   Fault Analyzer IDIS subsystem  
     usage 239  
   identification of fault entry 46  
   IDIXTSEL invocation exit 227  
   indication of invocation exit in ENV  
     data area 516  
   Notification user exit 402  
   process description 27  
   specifying user exits for 466

dump status  
   available 86  
   not found 86  
   suppressed 86  
 dump storage  
   locate with IDIXDLOC function 436  
 dump storage display example 147  
 dump storage suppressed display  
   example 149  
 dump suppression 15  
   controlling with user exit 395  
 DUMP tag 431  
 DUMP(No) 310  
 DUMPA tag 431  
 DumpDSN option 466  
 DumpRegistrationExits option 466  
 duplicate fault count 85  
 duplicate fault detection  
   overview 483  
 duplicate fault determination  
   controlling with user exit 393  
   criteria 490  
 duplicate fault processing  
   overview 30  
 duplicate history  
   showing 52  
 DUP\$ command 66  
 dynamic SVC update 232

## E

End Processing user exit 393  
 End Processing user exit (fault entry refresh) 396  
 Enterprise PL/I SYSDEBUG file  
   usage 288, 290  
 ENV data area 513  
 environment variable \_IDI\_OFF 357  
 EPC data area 521  
 EQAUEDAT exit  
   locating SYSDEBUG files 290  
 error conditions  
   application-handled  
     *See application-handled error conditions*  
   error display example 36  
 error handling in applications  
   effect on invocation of Fault Analyzer 229  
 ErrorHandler option 468  
 Evaluate REXX command 409  
 event details display example 115  
 event information  
   obtaining through IDIXEINF function 437  
 event summary 112  
 event summary display example 113  
 Exclude option 469  
 exclude processing  
   jobs analyzed with 281  
 exclude program from side file search display example 170  
 EXEC command 66  
 exit Fault Analyzer  
   file menu option 62  
 exit interactive reanalysis  
   file menu option 62

exits  
   CICS 227  
   invocation 223  
   Language Environment 224, 227  
   MVS 224, 227  
   user  
     *See user exits*  
 EXITS control statement 343  
 exits for invoking Fault Analyzer 223  
 Exits option 473

## F

FA exec 248  
 FA line command for ISPF 3.4 data set list 248  
 FA standard date format 512  
 fast Exclude options processing 282  
 fastpath navigation for CICS system  
   dump analysis 181  
 fault  
   applying action 52  
   reanalyzing 33  
 fault analysis report  
   viewing 52  
 Fault Analyzer  
   *See also* turning off (environment variable \_IDI\_OFF)  
   authorizing 231  
   CPU time consumed 407  
   customizing  
     preparation 217  
   customizing through IDIOPTLIM  
     configuration-options module 257  
   customizing through user exits 359  
   customizing through  
     USERMODS 253  
   displaying copyright information 80  
   displaying general usage information 80  
   enabling implicit invocation from PL/I V2R3 applications 255  
   enabling invocation 253  
   invoking  
     PL/I PLIDUMP 255  
     SDSF 249  
   Invoking  
     Java catch block 25  
     Java dump events 26  
   maintaining 353  
   migrating  
     from earlier version 211  
     from V10.1 to V11.1 212  
     from V11.1 to V12.1 212  
     from V12.1 to V13.1 211  
     from V6.1 to V7.1 214  
     from V7.1 to V8.1 213  
     from V8.1 to V9.1 213  
     from V9.1 to V10.1 213  
   registering in IFAPRDxx parmlib member 238  
   report 197  
   running with similar third-party products 228  
   stopping analysis 258  
   turning off (IFAPRDxx parmlib member) 355

Fault Analyzer (*continued*)  
   turning off (JCL switch IDIOFF) 356  
   uninstalling temporarily 355  
   verifying customization 325  
 Fault Analyzer client for IBM Rational Developer for System z 444  
 Fault Analyzer DB2 performance improving 317  
 Fault Analyzer ISPF interface 33  
 Fault Analyzer modules  
   authorizing 231  
   making available 231  
 Fault Analyzer options 139  
 Fault Analyzer options display example 140  
 Fault Analyzer plug-in for Eclipse 205, 443  
 Fault Analyzer preferences  
   options menu option 63  
 Fault Analyzer preferences display example 81  
 fault entries  
   finding 49  
   locking to prevent deletion 61  
   matching 47, 49  
   selecting 49  
   sending from one system to another 270  
   sorting 47, 49  
 fault entry duplicate history  
   viewing 87  
 fault entry duplicate history display example 88  
 fault entry expiration control 61  
 fault entry information  
   file menu option 62  
   refreshing 59  
   viewing 82  
 fault entry information display example 83  
 fault entry list column configuration display example 43  
 fault entry list display  
   customizing 283  
 fault entry list display example 35  
 fault entry list display with updates pending example 60  
 fault entry refresh  
   End Processing user exit 396  
 fault entry refresh processing 171  
 fault entry selection  
   specifying initial criteria 264  
 fault history entries  
   copying 90  
   cursor match 51  
   deleting 80  
   deleting many 82  
   deleting range 80  
   moving 91  
   moving range 91  
   transmitting 91  
   transmitting range 91  
 wildcard  
   wildcard 51  
 fault history files 8  
   selection during real-time execution 16

fault identifier  
     format 478  
 fault reanalysis 6  
     batch 95  
     creating your own job 101  
     interactive 103  
 FaultID option 478  
 File Browse display example 42  
 file information display example 118  
 File Manager  
     customizing DB2 environment 317  
     customizing for CICS 301  
     customizing for ISPF 247  
     invoking through Language Environment for CICS 302  
     ISPF data set allocations  
         required 247  
 FILES control statement 338  
 FIND command 67  
 FIND command differences between  
     display types 68  
 fixes, getting 563  
 FND area  
     clearing 307  
 format CICS auxiliary trace data set  
     file menu option 62  
 Format CICS Auxiliary Trace Data Set display example 187  
 format tags  
     ADDR (address) 428  
     AREA (area) 428  
     DATA (data) 429  
     DD (definition description) 429  
     DL (definition list) 429  
     DT (definition term) 430  
     DUMP (EBCDIC dump) 431  
     DUMPA (ASCII dump) 431  
     HP (highlighted phrase) 432  
     L (line) 432  
     LI (list item) 433  
     NOTEL (note list) 433  
     P (paragraph) 433  
     TH (table heading) 434  
     U (underline) 434  
     UL (unordered list) 435  
 formatted text  
     write to report with IDIXWRIT function 441  
 Formatting user exit 381  
 Formatting User Exit Selection List example 165  
 fragments, syntax diagrams xi

**G**  
 general report information 197  
 general usage information  
     displaying 80  
 GenerateSavedReport option 478  
 global environment data area (ENV) 362  
 global resource serialization 273  
 GRS 273

**H**  
 help text  
     adding 76  
     removing 76  
 hex-dumped storage display  
     example 137  
 hex-value column  
     hiding 141  
 High Level Assembler  
     *See assembler*  
 HistCols option 476, 479  
 history entries  
     fault  
         *See fault history entries*  
 history file access information  
     resetting 58  
 history file entries  
     *See also fault history entries*  
     creating 185  
     deleting 80  
     moving 91  
     moving range 91  
 history file fault entries  
     creating 189  
     XFACILIT resource class 274  
 history file properties  
     file menu option 62  
 history file properties display  
     example 53  
 history file selection  
     controlling with user exit 393  
 history file settings  
     restricting change 233  
 history file updates  
     controlling with user exit 394, 396  
 history file updates pending display  
     example 60  
 history files  
     accessing through non-ISPF interfaces 205, 446  
     allocating 261  
     contents 8  
     copying entry 52  
     default name 262  
     deleting entry 52  
     determining size 261  
     faults  
         *See fault history files*  
     initiating interactive reanalysis 52  
     managing access 273  
     managing across MVS systems  
         without shared DASD 266  
     managing with IDIUTIL 337  
     matching faults 49  
     moving entry 52  
     PDSE-managed 261  
     refreshing 59  
     selecting for display 37  
     setting name 262, 459  
     setting up 261  
     sharing across sysplex 216, 272  
     showing duplicate history information 52  
     submitting batch dump reanalysis 52  
     transmitting entry 52  
     viewing dump data set name 53

history files (*continued*)  
     viewing saved fault analysis report 52  
 HLASM  
     *See assembler*  
 Hogan  
     sample Formatting user exit 388  
 HP tag 432

**I**  
 IBM File Manager for z/OS  
     ISPF data set allocations  
         required 247  
 IBM Problem Determination Tools for z/OS Common Component Common Server  
     customizing 443  
 IBM Support Assistant, searching for problem resolution 561  
 identification of column configuration source 44  
 IDI\_SDUMP\_ACCESS  
     XFACILIT resource class 25, 235  
 IDIADATA attributes 296  
 IDIADATA data sets 508, 509  
 IDIADATA DD statement 18, 285  
 IDIADATA specification via DataSets option 457  
 IDIALLOC REXX command 411  
 IDIBOPT DDname 454  
 IDIBOxxx specification via DataSets option 457  
 IDICNF00 452  
 IDICNF00 data sets  
     dropping 460  
 IDICNF00 parmlib member 453  
 IDICNF00 user exits  
     dropping 475  
 IDICNFUM 451  
 IDICNFUM user-options module 452, 453  
 IDICNFxx  
     specifying alternative parmlib data set 258  
 IDICNFxx parmlib member 279  
     alternative data set name 279  
 IDICNFxx parmlib member suffix  
     &SYSCLONE MVS system symbol 279  
     SYSCLONE MVS system symbol 279  
 IDICZSVC 232  
 IDIDATST 298  
 IDIDTEST REXX command 415  
 IDIDOC  
     READ access requirement 234  
 IDIDOC specification via DataSets option 457  
 IDIDOxxx specification via DataSets option 457  
 IDIDSECT concatenation 156  
 IDIDSECT specification via DataSets option 457  
 IDIDSECTdsn REXX command 415  
 IDIEventInfo REXX command 416  
 IDIEXEC DDname 362, 467, 473, 496

IDIEXEC specification via DataSets option 457  
 IDIFREE REXX command 417  
 IDIGSVRJ sample member 443  
 IDIHIST data set 518  
 IDIHIST DD statement 262  
 IDIHIST specification via DataSets option 457  
 IDIHIST\_GROUP\_DSN XFACILIT resource class 274  
 IDIHIST\_USERID\_DSN XFACILIT resource class 274  
 IDIHUSR 349  
 IDILANGX messages 558 return codes 533  
 IDILANGX attributes 296  
 IDILANGX data sets 510, 511  
 IDILANGX DD statement 18  
 IDILANGX specification via DataSets option 457  
 IDILC attributes 296  
 IDILC data sets 509  
 IDILC DD statement 18  
 IDILC specification via DataSets option 457  
 IDILCOB attributes 296  
 IDILCOB data sets 509, 510  
 IDILCOB DD statement 18  
 IDILCOB specification via DataSets option 457  
 IDILCOBO attributes 296  
 IDILCOBO data sets 510  
 IDILCOBO DD statement 18  
 IDILCOBO specification via DataSets option 457  
 IDILEDS 305  
 IDILPLI attributes 296, 297  
 IDILPLI data sets 511  
 IDILPLI DD statement 18  
 IDILPLI specification via DataSets option 457  
 IDILPLIE data sets 511, 512  
 IDILPLIE DD statement 18  
 IDILPLIE specification via DataSets option 457  
 IDIMAPS specification via DataSets option 457  
 IDIModQry REXX command 418  
 IDINDFUE CICS NoDup(CICSTFAST) override assembler exit 310  
 IDIOFF DD statement 356  
 IDIOPTLM configuration-options module customizing Fault Analyzer 257  
 IDIOPTLM sample member 257  
 IDIOPTS DD statement 101  
 IDIOPTS DDname 451, 452  
 IDIOPTS options file 312, 454  
 IDIPANEX sample member 256  
 IDIPLT 302  
 IDIPLTD 302  
 IDIPLTS 302  
 IDIRegisterFaultEntry REXX command 418  
 IDIREPRT DD statement 18  
 IDIREPRT DUMMY allocation 20  
 IDIRFR\_TDUMP\_HLQ XFACILIT resource class 236  
 IDIRLOAD DDname CSECT mapping 297  
 IDIS subsystem 239 starting 241 stopping 244  
 IDIS subsystem requirements for DB2 243  
 IDIS subsystem requirements for Java 244  
 IDIS subsystem storage requirements 243  
 IDIS\$NDX sample member 9  
 IDISCICS sample member 304  
 IDISCMDS command table 247  
 IDISCNFU sample member 453  
 IDISDB2B sample member 333  
 IDISDB2X sample member 318  
 IDISFEAP sample member 569  
 IDISFECL sample member 569  
 IDISFEMA sample member 569  
 IDISFEMP sample member 569  
 IDISFEQP sample member 569  
 IDISFESK sample member 569  
 IDISHIST sample member 261  
 IDISISPF sample member 247  
 IDISJCTL skeleton member 250  
 IDISNAP 20, 516 assembler invocation example 24 COBOL invocation example 23 entry specifications 21 input parameter list 21 invocation 21 PL/I invocation example 23 return specifications 22  
 IDISPDPM sample member 255  
 IDISPLI sample member 255  
 IDISPLI USERMOD 255  
 IDISPLIA sample member 255  
 IDISPLIA USERMOD 255  
 IDISRC1 sample member 328  
 IDISROBT sample member 268  
 IDISTSOB sample member 270  
 IDISUFM1 sample member 384  
 IDISUFM2 sample member 386  
 IDISUFM3 sample member 427  
 IDISUFM4 sample member 388  
 IDISUFM5 sample member 392  
 IDISUFMX sample member 166  
 IDISUSI sample member 223  
 IDISUTL1 sample member 408  
 IDISVENU sample member 233  
 IDISVJPN sample member 234  
 IDISVKOR sample member 234  
 IDISWRAP 25  
 IDISXNFY sample member 267, 271  
 IDISXPLA sample member 363  
 IDISXPLB sample member 363  
 IDISXPLC sample member 363  
 IDISXPLP sample member 363  
 IDISYSDB attributes 296  
 IDISYSDB data sets 512  
 IDISYSDB DD statement 18  
 IDISYSDB specification via DataSets option 457  
 IDITABD sample member 255  
 IDITABD USERMOD 255  
 IDITABD USERMOD installation verifying 329  
 IDITABX sample member 253  
 IDITRACE write text line to 440  
 IDITRACE DD statement 292, 363, 411, 414, 415, 416, 417, 419, 423, 424, 453, 472, 490, 544  
 IDITRACE under CICS 307  
 IDIUTIL batch utility 337 return codes 533  
 IDIUTIL batch utility user exit samples 347  
 IDIUTIL Delete user exit 404  
 IDIUTIL Import user exit 403  
 IDIUTIL ListHF user exit 405  
 IDIVIEWS DDname 37  
 IDIVPASM sample member 325, 329  
 IDIVPBLE sample member 329  
 IDIVPC sample member 328  
 IDIVPCOB sample member 326, 329  
 IDIVPDB2 sample member 331  
 IDIVPDBB sample member 333  
 IDIVPPLE sample member 327  
 IDIVPPLI sample member 327, 329  
 IDIVSEN  
     READ access requirement 234  
 IDIVSJPN  
     READ access requirement 234  
 IDIVSKOR  
     READ access requirement 234  
 IDIVSxxx specification via DataSets option 457  
 IDIWCIDI sample member 445  
 IDIWRITE REXX command 420  
 IDIWTO REXX command 421  
 IDIWSEL sample member 313  
 IDIXCCEE 302, 516  
 IDIXCEE 516  
 IDIXCEE LE exit enablement verifying 329  
 IDIXCX52 302, 516  
 IDIXCX53 302, 516  
 IDIXDCAP 516 installing 253  
 IDIXDCAP real-time analysis suppressing 256  
 IDIXDLOC function 436  
 IDIXEINF function 437  
 IDIXFA 302, 305  
 IDIXFXIT  
     entry specifications 276  
     example 277  
     input parameter list 277  
     return specifications 277  
     user exit purpose 276  
 IDIXGETN function 438  
 IDIXJAVA 25  
 IDIXLIST function 439  
 IDIXMAP 302  
 IDIXSFOR exit compiler listing 292 side file 292  
 IDIXSFOR input parameter list 293  
 IDIXTSEL 516  
 IDIXUFMT load module 435

IDIXUFMT load module (*continued*)  
     functions 436  
 IDIXDLOC function 436  
 IDIXEINF function 437  
 IDIXGETN function 438  
 IDIXGETS function 438  
 IDIXGETX function 439  
 IDIXLIST function 439  
 IDIXNOTE function 440  
 IDIXTRCE function 440  
 IDIXWRIT function 441  
 IDIXWTO function 441  
 IDIXWRT function 441  
 IDIXWTO function 441  
 IEATDUMP  
     changing default name for recovery fault recording 258  
 IEATDUMP dump title 32  
 IEFUSI exit sample 223  
 IFAPRDxx parmlib member 355  
     registering Fault Analyzer 238  
 IGZIUXB exit  
     locating SYSDEBUG files 290  
 implicit refresh 59  
 IMPORT control statement 342  
 IMS  
     LE considerations 319  
 IMS environment  
     customizing 319  
 IMS fast duplicate fault suppression 484  
 IMS information  
     system-wide information 131  
 IMS information display example 131  
 IMS information summary display  
     example 134  
 Include option 469  
 INFO command 69  
 information centers, searching for problem resolution 561  
 input parameter list  
     IDISNAP 21  
 input parameter list for IDIXSFOR 293  
 installation checklist 217  
 installation-wide default options 279, 453  
 interactive displays  
     copying to file 79  
 interactive options line 104  
 interactive reanalysis 7  
     data sets 171  
     initiating 52, 108  
     options 103  
     performing 103  
     performing under CICS 206  
 interactive reanalysis options  
     fault entry list display options menu option 63  
 interactive reanalysis options display  
     example 104  
 interactive reanalysis report display  
     example 110  
 interactive reanalysis status display  
     example 109  
 interactive reanalysis under CICS 445  
 interactive report  
     abend job information 139  
     COBOL Explorer 172

interactive report (*continued*)  
     deferred breakpoints feature 176  
     detailed event information 114  
     displaying associated storage areas 140  
     displaying Java analysis 191  
     displaying source code 145  
     displaying storage locations 147  
     event summary 112  
     expanding abend codes 144  
     expanding messages 144  
     Fault Analyzer options 139  
     general information 109  
     synopsis 112  
     user notes 139  
 InteractiveExitPromptSeconds option 477, 479  
 Internet  
     searching for problem resolution 561  
 introduction 3  
 invocation for CICS transaction abends 227  
 IPL requirement 232  
 ISPF  
     verifying Fault Analyzer 335  
 ISPF data sets  
     allocating for Japanese feature 321  
     allocating for Korean feature 323  
 ISPF environment  
     modifying 247  
 ISPF interface 33  
     commands 64  
     customizing 283  
     invoking 34  
     on-line help 34  
     providing defaults for new users 250  
     security considerations 92  
 ISPF packed data format 299  
 ISPF selection panel  
     updating 248  
 ISPF split screen support 34  
 ISPLIBD 247  
 ISRDDN 247  
 ISRFIND 247  
 IVP testing  
     CICS 329

## J

Japanese feature  
     allocating ISPF data sets 321  
     customizing 321  
 Java  
     IDIS subsystem requirements 244  
     invoking Fault Analyzer 25  
 Java abend analysis  
     selecting dump data set 189  
 Java analysis 189  
     displaying in interactive report 191  
     setting options 189  
 Java API  
     download report 581  
     download reports 573  
 Java application abends  
     required LE options 228

Java capture SDUMP data sets  
     managing with XFACILIT resource class 25  
 Java dump analysis  
     creating history file fault entry 189  
 Java dump data set  
     selecting 189  
 Java dump events  
     Invoking Fault Analyzer 26  
 Java event details display example 194  
 Java event summary display  
     example 193  
 Java fault entry reanalysis 190  
 Java information  
     system-wide information 137  
 Java information display example 195  
 Java interactive reanalysis report display  
     example 192

## K

keywords, syntax diagrams xi  
 knowledge bases, searching for problem resolution 561  
 Korean feature  
     allocating ISPF data sets 323  
     customizing 323

## L

L tag 432  
 Language Environment  
     *See LE*  
 Language Environment abnormal termination exit for CICS enabling 302  
 Language Environment for CICS  
     configuring to invoke Fault Analyzer 302  
 Language Environment heap analysis  
     system-wide information 137  
 Language Environment heap analysis information display example 138  
 Language Environment options required for invocation of Fault Analyzer 228  
 Language option 479  
 last accessed history file entries  
     file menu option 62  
 last accessed history file entries display  
     example 41  
 last accessed history files or views  
     file menu option 62  
 last accessed history files or views display example 40  
 last CICS 3270 screen buffer  
     system-wide information 120  
 last CICS 3270 screen buffer display  
     example 121  
 last CICS 3270 screen buffer hex  
     system-wide information 121  
 last CICS 3270 screen buffer hex display  
     example 122  
 LE  
     DB2 considerations 317  
     IMS considerations 319  
     not in LINKLIST 257

LE (*continued*)  
     preventing wrap of CICS trace 312  
 LE abnormal termination exit  
     enabling 254  
     MVS change options/suppress dump  
         exit 225  
 LE options 228, 312  
 LE options required for CICS  
     abends 228  
 LE options required for non-CICS  
     abends 228  
 LE options required to capture Java  
     application abends 228  
 LE parameter list  
     non-standard separator character 256  
 LE runtime library  
     identifying 257  
 level 88 items  
     collapsing 142  
 level 88 items display example 143  
 LI tag 433  
 library names after you finish  
     installing 221  
 license inquiry 583  
 LINKLIST  
     modules in 231  
 List REXX command 422  
 list user notes  
     services menu option 63  
 list views  
     file menu option 62  
 LISTHF control statement 338  
 listing/side file mismatch display  
     example 169  
 listings  
     pointing to 283  
 load modules  
     obtaining from CA-Panexec 256  
 LOADER restriction 15  
 locale name 512  
 Locale option 480  
 lock flag 61, 84, 519  
 LOOKC command 70  
 LOOKC ISPF command 250  
 lookup  
     services menu option 63  
 LOOKUP command 69  
     invoking through cursor  
         selection 250  
 Lookup Search and Browse display  
     example 78  
 LoopProtection option 481  
 LPA  
     modules in 232  
     placing modules in 353  
 LPA module compatibility 216  
 LST data area 522

## M

main report sections 199  
 MATCH ALL match condition 51  
 MATCH command 49, 51, 70  
 MATCH CSR match condition 51  
 maximum call depth 202  
 MaxMinidumpPages option 482

menus  
     action-bar pull-down 61  
 Message and Abend Code Explanation  
     user exit 377  
 message explanation display  
     example 144  
 message explanation repository  
     setting up 233  
 message explanations  
     user-defined 349  
 message id look-up display example 78  
 messages 535  
     expanding 144  
     Fault Analyzer 535  
     IDILANGX 558  
     system-wide information 128  
 minidumps  
     concept 3  
     limiting size 282  
 minimum storage requirements 222  
 modules  
     *See* Fault Analyzer modules  
 MTRACE records  
     system-wide information 138  
 MTRACE Records display example 138  
 multicultural support 321, 323, 479, 480  
 MVS change options  
     installing 253  
 MVS change options/suppress dump exit  
     LE abnormal termination exit 225  
 MVS console  
     use IDIXWTO function to write  
         to 441  
 MVS dump data set size 229  
 MVS post-dump exit IDIXTSEL  
     installing 313  
 MVS SVC dump registration  
     Analysis Control user exit 371  
     Notification user exit 402

## N

national language  
     setting 321, 323  
 new history file allocation  
     file menu option 62  
 new history file allocation display  
     example 56  
 NEXT command 70  
 NFY data area 524  
 NODUMP  
     EXEC CICS ABEND command 305  
 NODUMP keyword on EXEC CICS  
     ABEND command 305  
 NoDup option 482  
 NoDup(CICSFAST) option  
     changing 489  
 NoDup(ImageFast) option  
     changing 486  
 NoErrorHandler option 468  
 non-CICS transaction abends  
     invoking analysis 223  
 non-ISPF interfaces  
     accessing history files 205, 446  
     installing 446

non-ISPF interfaces to access Fault  
     Analyzer history files  
         installing 443  
 non-LE run time  
     applications 255  
 non-REXX logging routine  
     calling from REXX 398  
 non-REXX user exit buffered data  
     format 505  
 NoPrintInactiveCOBOL option 494  
 NoQuiet option 495  
 NoSource option 499  
 NoSpinIDIREPRT option 499  
 not found dump status 86  
 Note REXX command 423  
 NOTEL tag 433  
 NOTELIST command 71  
 Notification user exit 397  
 Notification user exit (MVS SVC dump  
     registration) 402  
 nteractive report  
     exit 111

## O

object code  
     disassembling with DISASM  
         command 160  
 ON ERROR  
     usage considerations 229  
 open files  
     system-wide information 117  
 operating environment  
     customizing 231  
 options  
     \_IDI\_OPTS environment variable 455  
     \_IDI\_OPTSFILE environment  
         variable 454  
     AdditionalIDIOffDD 455  
     batch reanalysis 95  
     CICSDumpTableExclude 456  
     CICSTraceMax 456  
     cumulative 452  
     DataSets 283, 457  
     default  
         *See* default options  
     DeferredReport 463  
     Detail 282, 465  
     DumpDSN 466  
     DumpRegistrationExits 466  
     ErrorHandler 468  
     Exclude 281, 469  
     Exits 473  
     FaultID 478  
     general description 451  
     GenerateSavedReport 478  
     HistCols 476, 479  
     IDICNF00 parmlib member 453  
     IDIOPTS 454  
     Include 281, 469  
     installation-wide 453  
     interactive reanalysis 103  
     InteractiveExitPromptSeconds 477,  
         479  
     Language 321, 323, 479  
     Locale 480  
     LoopProtection 481

options (*continued*)  
     MaxMinidumpPages 282, 482  
     NoDup 482  
     NoErrorHandler 468  
     NoPrintInactiveCOBOL 494  
     NoQuiet 495  
     NoSource 499  
     NoSpinIDIREPRT 499  
     PARM field 455  
     PDTCCopts 491  
     PermitLangx 491  
     PreferredFormattingWidth 494  
     PrintInactiveCOBOL 494  
     purpose 451  
     Quiet 283, 495  
     RDZClient 495  
     RefreshExits 496  
     RetainCICSDump 497  
     RetainDump 498  
     setting 321  
     setting for CICS system abend analysis 177  
     setting for Java analysis 189  
     Source 499  
     SpinIDIREPRT 499  
     StoragePrintLimit 499  
     StorageRange 501  
     syntax rules 452  
     SystemWidePreferred 502  
     UseIDISTime 503  
     user options file 454  
     user-options module IDICNFUM 453  
     WDZClient 495  
     where set 451  
     where to specify 452  
 options in effect 184  
 options in effect display example 185  
 options menu  
     deletion options 81  
 organization of this book xi  
 over-typing existing values 50

## P

P tag 433  
 packed data format  
     ISPF  
         *See* ISPF packed data format  
 PARM field options 455  
 parmlib data set  
     specifying alternative for  
         IDICNFxx 258  
 PDS  
     allocating as history file 261  
 PDSE  
     allocating as history file 261  
 PDSE-managed history files 261  
 PDTCCopts option 491  
 PermitLangx option 491  
 PF keys  
     showing 35  
 PL/I  
     verifying Fault Analyzer 327  
 PL/I invocation  
     IDISNAP 23  
 PL/I ON ERROR  
     usage consideration 229

PL/I V2R3 applications  
     enabling implicit Fault Analyzer invocation 255  
 PL/I version 2 release 3  
     USERMOD to facilitate invocation of Fault Analyzer 255  
 PLIDUMP  
     invoking Fault Analyzer 255  
 point of failure 522  
 point-and-shoot fields 110  
 preferred formatting width  
     view menu option 63  
 preferred formatting width display example 77  
 PreferredFormattingWidth option 494  
 PREV command 71  
 PrintInactiveCOBOL option 494  
 problem determination  
     describing problems 565  
     determining business impact 565  
     submitting problems 566  
 product enablement 238  
 program control access  
     defining 232  
 program descriptions  
     user-defined 351  
 program SNAP interface 20  
 programs  
     setting up for fault analysis 12  
 prompting  
     controlling 171  
 pull-down menus 61

## Q

Quiet option 495  
 QUIT command 71

## R

RDZClient option 495  
 READ access requirement  
     IDIDOC 234  
     IDIVSENU 234  
     IDIVSJPN 234  
     IDIVSKOR 234  
 real-time abend analysis 3  
 real-time analysis 15  
     controlling with options 17  
 real-time analysis report 18  
 real-time fault analysis report  
     viewing 74  
 real-time reports  
     combining 19  
     controlling SYSOUT class 19  
     suppressing 19  
 real-time SNAP analysis 5  
 reanalysis  
     batch 52, 95  
         *See* batch reanalysis  
     interactive 103  
         *See* interactive reanalysis  
 recording area  
     clearing NoDup(CICSFAST(...)) 307  
 recovery fault recording  
     *See also* RFR

recovery fault recording (*continued*)  
     changing default IEATDUMP data set name 258  
     Fault Analyzer IDIS subsystem usage 239  
     overview 30  
     verifying set-up 335  
 recovery fault recording data set access managing 234  
 refresh  
     view menu option 64  
 REFRESH command 59, 72  
 refresh processing 171  
 RefreshExits option 496  
 region size  
     extra required 12  
 region size requirements 222  
 remove blank lines  
     view menu option 64  
 remove help text  
     view menu option 64  
 remove pseudo-assembler instructions  
     view menu option 64  
 repeatable items, syntax diagrams xi  
 report  
     contents 197  
 report detail  
     controlling 282  
 report examples 204  
 report sections  
     abend job information 204  
     epilog 204  
     event details 202  
     main 199  
     options 204  
     prolog 200  
     summary 200  
     synopsis 200  
     system-wide information 203  
 report user exit  
     *See* Analysis Control user exit  
 reports  
     download with Java API 573, 581  
     increasing display area 251  
     real-time  
         *See* real-time reports  
 required programs  
     defining to CICS 302  
 RESET command 72  
 RetainCICSDump option 497  
 RetainDump option 498  
 return codes from batch reanalysis 533  
 return codes from IDILANGX 533  
 return codes from IDIUTIL batch utility 533  
 return specifications  
     IDISNAP 22  
 REXX  
     calling non-REXX logging routine 398  
 REXX commands 409  
     Evaluate 409  
     IDIALLOC 411  
     IDIDDTEST 415  
     IDIDSECTdsn 415  
     IDIEventInfo 416  
     IDIFREE 417

REXX commands (*continued*)  
 IDIModQry 418  
 IDIRRegisterFaultEntry 418  
 IDIWRITE 420  
 IDIWTO 421  
 List 422  
 Note 423  
 REXX exec libraries  
     pointing to 283  
 REXX SAY instruction 367  
 REXX TRACE instruction 367  
 RFR dump titles 32  
 RFR SDUMP data sets  
     XFACILIT resource class 235  
 RFR TDUMP XFACILIT example 237  
 routine  
     *See non-REXX logging routine*  
 routing codes  
     WTO messages 13  
 row count 36, 113  
 RPTFIND command 72  
 RUNCHAIN command 73

## S

sample CICS definition job 304  
 sample customized ISPF interface  
     front-end 569  
 sample data set members  
     IDICNFxx 280  
     IDIGSVRJ 443  
     IDIOPTLM 257  
     IDIPANEX 256  
     IDIS\$NDX 9  
     IDISCICS 304  
     IDISCNFU 453  
     IDISDB2B 333  
     IDISDB2X 318  
     IDISFEAP 569  
     IDISFECL 569  
     IDISFEMA 569  
     IDISFEMP 569  
     IDISFEQP 569  
     IDISFESK 569  
     IDISHIST 261  
     IDISISPF 247  
     IDISPDM 255  
     IDISPLI 255  
     IDISPLIA 255  
     IDISRC1 328  
     IDISROBT 268  
     IDISTSOB 270  
     IDISUFM1 384  
     IDISUFM2 386  
     IDISUFM3 427  
     IDISUFM4 388  
     IDISUFM5 392  
     IDISUFMX 166  
     IDISUSI 223  
     IDISUTL1 408  
     IDISVENU 233  
     IDISVJPN 234  
     IDISVKOR 234  
     IDISXNFY 267  
     IDISXPLA 363  
     IDISXPLB 363  
     IDISXPLC 363

sample data set members (*continued*)  
     IDISXPLP 363  
     IDITABD 255  
     IDITABX 253  
     IDIVPASM 325, 329  
     IDIVPBLE 329  
     IDIVPC 328  
     IDIVPCOB 326, 329  
     IDIVPDB2 331  
     IDIVPDBB 333  
     IDIVPPLE 327  
     IDIVPPLI 327, 329  
     IDIWCIDI 445  
     IDIWTSEL 313  
     IDIXMIT 271  
 sample reports 204  
 sample XFACILIT implementation 275, 276  
 saved report  
     viewing 74  
 saved report display example 75  
 SAY  
     REXX instruction 367  
 SDUMP recovery fault recording data sets 235  
 SDUMP SVC  
     screening 304  
 security considerations 92  
 SETFAULTPREFIX control statement 340  
 SETMAXFAULTENTRIES control  
     statement 341  
 SETMINFAULTENTRIES control  
     statement 342  
 SETPROG command  
     managing modules 353  
 SFA command 249  
 SHOW command 73  
 shutdown PLT  
     adding required programs 303  
 side file availability test utility 298  
 side files  
     advantage over listings 12  
     attributes 296  
     locating 290  
     naming 289  
     prompting display 166  
     providing 285  
     selection criteria 12  
     temporary data set 297  
 SLIP TRAP  
     effect on real-time analysis 224  
 SLIP traps 229  
 SLIP,COMP=0C4 trap 32  
 SMF type 89 record 29, 355  
 Software Support  
     contacting 564  
     describing problems 565  
     determining business impact 565  
     receiving updates 563  
     submitting problems 566  
 source code  
     displaying 145  
 Source option 499  
 special history file data set members 9  
     \$\$BACKUP 9  
     \$\$INDEX 9  
 special processing of Language Environment CEEWUCHA abends 13  
 specify compiler listing or side file  
     display example 168  
 specify move/copy options display  
     example 90  
 specify XMIT options display  
     example 92  
 SpinIDIREPRT option 499  
 SQLCA 131, 502  
 startup PLT  
     adding required programs 303  
 status pop-up display 108  
 STCK command 74  
 STCK conversion display example 163  
 storage areas  
     displaying associated 140  
     mapping with DSECT  
         information 154  
         report with IDIXLIST function 439  
         system-wide information 136  
 Storage Disassemble display  
     example 161  
 storage DSECT mapping entry display  
     example 155  
 storage DSECT mapping map display  
     example 156  
 storage locations  
     displaying 147  
 storage recommendations 222  
 storage requirements 222  
     CICS 313  
 storage RUNCHAIN command entry  
     display example 158  
 StoragePrintLimit option 499  
 StorageRange option 501  
 STORE CLOCK values (STCK command)  
     converting 162  
 substitution symbols in data set  
     names 369, 460  
 subsystem for Fault Analyzer  
     user exits running from 467  
 subsystems  
     IDIS 239  
 successful analysis  
     criteria 522  
 summarized CICS trace  
     system-wide information 122  
 summarized CICS trace display  
     example 123  
 summary of exit usage 225  
 supported application environments 9  
 suppress dump exit IDIXDCAP  
     installing 253  
 suppressed copybooks 199  
 suppressed dump status 86  
 SVC 51  
     screening 304  
 SVC dump exit 313  
 SVC dump registration exit 227  
 SVC dump screening 304  
 SVC installation  
     performing dynamically with  
         IDICZSVC 232  
 SVCDUMP dump title 32  
 symbol substring specification 462  
 symbols in data set names 369, 460

synopsis 112, 181  
 synopsis display example 112  
 syntax diagrams, how to read xi  
 SYSADATA DD statement 285  
 SYSCLOSE MVS system symbol  
     IDICNFxx parmlib member  
     suffix 279  
 SYSDEBUG files  
     locating with EQAUEDAT exit 290  
     locating with IGZIUXB exit 290  
 syslog messages  
     suppressing noncritical 283  
 SYSLOG summary 20  
 SYSMDUMP  
     ASA printer control characters 466  
     logical record length 466  
     specifying as input to batch  
     reanalysis 466  
 SYSMDUMP open prompt 105  
 SYSOUT class of real-time reports  
     controlling 19  
 sysplex  
     sharing history files 216, 272  
 Sysplex-wide subsystem  
     inter-communication 240  
 system dump suppression  
     controlling with user exit 395  
 system-wide information  
     CICS 119  
     CICS channels 126  
     CICS commareas 126  
     CICS information  
         CICS trace formatting 123  
         last CICS 3270 screen buffer 120  
         last CICS 3270 screen buffer  
             hex 121  
             summarized CICS trace 122  
     CICS levels 126  
     DB2 information 129  
     IMS information 131  
     Java Information 137  
     Language Environment heap  
         analysis 137  
     messages 128  
     MTRACE records 138  
     open files 117  
     storage areas 136  
 system-wide messages display  
     example 129  
 system-wide open files display  
     example 117  
 system-wide storage areas display  
     example 136  
 SystemWidePreferred option 502  
 SYSUDUMP SYSOUT class  
     allocation of IDIREPRT 18

**T**

table displays  
     matching 183  
     sorting 183  
 TACB  
     duplicate determination under  
         CICS 488  
 tags  
     *See* formatting tags

TDUMP recovery fault recording data  
     sets 236  
 temporary data sets  
     compiler listing 297  
     side file 297  
 termination  
     ensuring correct 303  
 TEST option considerations 288  
 TEST(NONE,SYM,SEPARATE) COBOL  
     compiler option 288, 290  
     restriction with display of  
         pseudo-assembler instructions 146  
 TEST(STMT,SYM,NOHOOK,SEPARATE)  
     Enterprise PL/I compiler option 288,  
     290  
 TH tag 434  
 third-party products 228  
 title 73, 571  
 total rows displayed 36, 113  
 TRACE  
     REXX instruction 367  
 transaction abend analysis  
     CICS  
         *See* CICS  
         ensuring not suppressed by  
             DUMP(NO) 310  
 transaction abend control block  
     *See* TACB  
 TSO  
     REXX environment restrictions 362

## U

U tag 434  
 UFM data area 525  
 UL tag 435  
 updates pending 59  
 UseIDISTime option 503  
 user exit type specific data area 362  
 user exits 359  
     Analysis Control 368  
     Analysis Control (MVS SVC dump  
         registration) 371  
     CICS system abend analysis 177  
     Compiler Listing Read 372  
     data area version checking 363  
     dropping 475  
     End Processing 393  
     End Processing (fault entry  
         refresh) 396  
     Formatting 381  
     IDIUTIL Delete 404  
     IDIUTIL Import 403  
     IDIUTIL ListHF 405  
     invocation parameters 362  
     Message and Abend Code  
         Explanation 377  
     Notification 397  
     Notification (MVS SVC dump  
         registration) 402  
     substitution symbols in data set  
         names returned 369  
     supported programming  
         languages 362  
     tracing 363  
 user fields update prompt example 87

user modifications  
     parmlib data set name 279  
 user note list display example 153  
 user notes 139  
     creating 150  
     managing 150  
 user notes update prompt example 154  
 user options file 454  
 user-defined  
     abend code explanations 350  
     message explanations 349  
     program descriptions 351  
 user-options file 451  
 user-options module 451  
 user-options module IDICNFUM 453  
 user-selected message explanations  
     displaying 77  
 user-specific report formatting (EXEC  
     command) 164  
 USERMODS  
     IDISPLI 255  
     IDISPLIA 255  
     IDITABD 255  
     IDITABX 253  
     IDIWTSEL 313  
     re-applying 353  
     restoring 353  
 USRHDLR(CEEWUCHA)  
     special processing 13  
 UTL data area 530

## V

values  
     matching 51  
     matching with cursor-selecting 49  
 variables, syntax diagrams xi  
 view considerations  
     without XFACILIT resource class 265  
 view list display example 42  
 views 37  
     selecting for display 37  
     setting up 263  
 VIEWS command 74

## W

WDZClient option 495  
 web interface 206  
     installing 446  
 WebSphere or Java dump analysis 189  
 width  
     setting preferred formatting 76  
 wildcards  
     matching fault history entries 51  
 write-to-operator messages  
     *See* WTO  
 WTO  
     descriptor codes 13  
     routing codes 13

## X

XCF 240, 273  
 XFACILIT implementation example 275,  
     276

XFACILIT resource classes  
  IDI\_SDUMP\_ACCESS 25, 235  
  IDIHIST\_GROUP\_DSN 274  
  IDIHIST\_USERID\_DSN 274  
  IDIRFR\_TDUMP\_HLQ 236  
managing history file fault  
  entries 274  
managing Java capture SDUMP data  
  sets 25  
managing recovery fault recording  
  SDUMP data sets 235  
managing TDUMP RFR data sets 236  
XPL data area 530



**IBM**<sup>®</sup>

Printed in USA

SC19-4116-10

