

IBM Fault Analyzer for z/OS



# User's Guide and Reference

*Version 12 Release 1*

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 569.

**First Edition (May 2012)**

This edition applies to Version 12 Release 1 Modification Level 0 of IBM Fault Analyzer for z/OS, Program Number 5655-W69, 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 may be downloaded from MVS™ to your PC in BINARY mode from IDI.SIDIDOC1(IDIUGPDF)<sup>1</sup> for viewing and printing.

For the Japanese feature of Fault Analyzer, the most recently translated Japanese edition is available in IDI.SIDIDJPN(IDIUGPDF)<sup>1</sup>.

For the Korean feature of Fault Analyzer, the most recently translated Korean edition is available in IDI.SIDIDKOR(IDIUGPDF)<sup>1</sup>.

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

© Copyright IBM Corporation 2000,2012.

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

---

1. This data set is available on the mainframe MVS system on which Fault Analyzer is installed.

---

# Contents

## About this book . . . . . xi

Who should use this book . . . . . xi

Organization of this book . . . . . xi

How to read the syntax diagrams . . . . . xi

## How to send your comments to IBM . . . xv

If you have a technical problem . . . . . xv

## Summary of changes . . . . . xvii

IBM Fault Analyzer for z/OS Version 9 Release 1 . . . . . xvii

SC19-2538-00: September 2008. . . . . xvii

SC19-2538-01: APAR PK74751 edition (January 2009) . . . . . xvii

SC19-2538-02: APAR PK79442 edition (March 2009) . . . . . xviii

SC19-2538-03: APAR PK89333 edition (July 2009) . . . . . xix

SC19-2538-04: APAR PK92822 edition (September 2009) . . . . . xx

IBM Fault Analyzer for z/OS Version 10 Release 1 . . . . . xx

SC19-2868-00: November 2009 . . . . . xx

SC19-2868-01: APAR PM00092 edition (December 2009) . . . . . xxi

SC19-2868-02: APAR PM08932 edition (April 2010) . . . . . xxi

SC19-2868-03: APAR PM11905 edition (May 2010) . . . . . xxii

SC19-2868-04: APAR PM14420 edition (August 2010) . . . . . xxii

SC19-2868-05: APAR PM21096 edition (October 2010) . . . . . xxii

SC19-2868-06: APAR PM27330 edition (February 2011) . . . . . xxiii

IBM Fault Analyzer for z/OS Version 11 Release 1 . . . . . xxiii

SC19-3131-00: November 2010 . . . . . xxiii

SC19-3131-01: APAR PM28208 Edition (February 2011) . . . . . xxiii

SC19-3131-02: APAR PM34295 Edition (June 2011) . . . . . xxiv

SC19-3131-03: APAR PM44693 Edition (September 2011) . . . . . xxv

SC19-3131-04: APAR PM47919 Edition (December 2011) . . . . . xxv

SC19-3131-05: APAR PM54422 Edition (February 2012) . . . . . xxv

IBM Fault Analyzer for z/OS Version 12 Release 1 . . . . . xxvi

SC19-3671-00: May 2012 . . . . . xxvi

---

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

### Chapter 1. Introduction . . . . . 3

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 . . . . . 10

Setting up existing programs for fault analysis . . 10

Additional region size required . . . . . 11

Compiler listing or side file selection criteria . . 11

Special processing of Language Environment

CEEWUCHA abends . . . . . 11

WTO routing and descriptor codes used by Fault

Analyzer . . . . . 12

### Chapter 2. Real-time analysis . . . . . 13

Dump suppression . . . . . 13

Fault Analyzer and CICS global user exits . . . 14

Fault history file selection . . . . . 14

Controlling the real-time analysis with options . . 15

Pointing to listings with JCL DD statements . . . 16

The real-time analysis report . . . . . 16

Combining Fault Analyzer real-time reports . . 17

Controlling the SYSOUT class of real-time reports 17

Suppressing real-time reports . . . . . 17

The SYSLOG summary . . . . . 18

Using the program SNAP interface (IDISNAP) . . 18

IDISNAP invocation . . . . . 18

Invoking Fault Analyzer from Java catch block . . 22

Invoking Fault Analyzer from Java dump events . . 24

Dump registration processing . . . . . 24

Real-time exclusion processing . . . . . 25

Duplicate fault processing . . . . . 28

Recovery fault recording . . . . . 28

RFR dump titles . . . . . 30

Using SLIP,COMP=0C4 with Fault Analyzer . . . 30

### Chapter 3. The Fault Analyzer ISPF interface . . . . . 31

Invoking the interface . . . . . 32

ISPF split screen support . . . . . 32

The Fault Entry List display . . . . . 32

Using views . . . . . 35

Changing the history file or view displayed . . 35

Fault entry list column configuration . . . . . 41

Sorting and matching fault entries . . . . . 45

Additional ways to match and select faults . . . 46

Applying an action to a particular fault . . . . 49

History file properties . . . . . 51

New history file allocation . . . . . 53

Change fault history file settings . . . . . 54

Resetting history file access information . . . . 56

Refreshing fault entry information . . . . . 56

Fault entry expiration control . . . . . 58

Action-bar pull-down menus . . . . . 58

Commands . . . . . 61

COLS . . . . . 61

COPY . . . . .	62
DISASM . . . . .	62
DSECT . . . . .	62
DUPS . . . . .	63
EXEC . . . . .	63
FIND . . . . .	63
INFO . . . . .	65
LOOKUP . . . . .	66
MATCH . . . . .	66
NEXT . . . . .	67
NOTELIST . . . . .	67
PREV . . . . .	67
QUIT . . . . .	68
REFRESH . . . . .	68
RESET . . . . .	68
RPTFIND . . . . .	68
RUNCHAIN . . . . .	69
SHOW . . . . .	69
STCK . . . . .	70
VIEWS . . . . .	70
Viewing a saved report . . . . .	70
Adding or removing blank lines . . . . .	72
Adding or removing help text . . . . .	72
Setting preferred formatting width . . . . .	72
Displaying user-selected message or abend code explanations . . . . .	73
Copying interactive displays to a file . . . . .	75
Displaying Fault Analyzer copyright and general usage information . . . . .	76
Deleting history file entries . . . . .	76
Changing deletion options through the Options menu . . . . .	77
Deleting when the confirmation display is shown . . . . .	77
Deleting when the confirmation display is not shown . . . . .	78
Bulk deletions . . . . .	78
Viewing fault entry information . . . . .	78
Viewing the fault entry duplicate history . . . . .	83
Copying history file entries . . . . .	86
Moving history file entries . . . . .	87
Transmitting history file entries . . . . .	87
Security considerations . . . . .	88

## Chapter 4. Performing batch reanalysis 89

Batch reanalysis options . . . . .	89
Initiating batch reanalysis. . . . .	94
Data sets used for batch reanalysis . . . . .	94
Creating your own batch reanalysis job . . . . .	95

## Chapter 5. Performing interactive reanalysis . . . . . 97

Interactive reanalysis options . . . . .	97
Initiating interactive reanalysis . . . . .	101
Status pop-up display . . . . .	101
General information about the interactive report . . . . .	102
Exit from the interactive report . . . . .	104
Primary option: Synopsis . . . . .	105
Primary option: Event Summary . . . . .	105
Detailed Event Information. . . . .	107
Primary option: Open Files . . . . .	110

Primary option: CICS Information . . . . .	112
Last CICS 3270 Screen Buffer . . . . .	113
Last CICS 3270 Screen Buffer Hex . . . . .	113
Summarized CICS Trace. . . . .	114
CICS Trace Formatting . . . . .	115
CICS Levels, Commareas, and Channels . . . . .	117
Primary option: Messages . . . . .	120
Primary option: DB2 Information. . . . .	120
Primary option: IMS Information. . . . .	123
Primary option: Storage Areas. . . . .	126
Primary option: Java Information. . . . .	127
Primary option: WebSphere Information . . . . .	127
Primary option: Language Environment Heap Analysis . . . . .	127
Primary option: MTRACE Records . . . . .	128
Primary option: Abend Job Information . . . . .	129
Primary option: User Notes . . . . .	129
Primary option: Fault Analyzer Options . . . . .	129
Displaying associated storage areas . . . . .	130
Hiding the hex-value column . . . . .	131
Collapsing level 88 items . . . . .	132
Expanding messages and abend codes . . . . .	134
Displaying source code . . . . .	135
Displaying storage locations . . . . .	138
Creating and managing user notes . . . . .	140
Mapping storage areas using DSECT information . . . . .	145
IDIDSECT concatenation . . . . .	147
Indexing your DSECT data sets (\$DINDEX member) . . . . .	147
DSECT indexing utility (IDIPDSCU). . . . .	148
Displaying chained data areas. . . . .	148
Disassembling object code . . . . .	150
Converting STORE CLOCK values . . . . .	152
User-specific report formatting . . . . .	154
Prompting for compiler listing or side file. . . . .	156
Controlling prompting . . . . .	160
Data sets used for interactive reanalysis . . . . .	160
Refresh processing. . . . .	161

## Chapter 6. Performing CICS system abend dump analysis . . . . . 163

Setting options for CICS system abend analysis . . . . .	163
User exits . . . . .	163
Selecting a CICS dump data set . . . . .	163
Selecting an address space to analyze . . . . .	165
Displaying the CICS system abend interactive report . . . . .	166
Fastpath navigation . . . . .	167
Option 1: Synopsis . . . . .	167
Option 2: Abend Job Information. . . . .	167
Option 3: CICS System Information . . . . .	168
Option 4: Options in Effect . . . . .	170
Creating a history file entry . . . . .	171

## Chapter 7. Formatting a CICS auxiliary trace data set . . . . . 173

Selecting a CICS auxiliary trace data set . . . . .	173
Specifying CICS Trace Selection Parameters . . . . .	174

## Chapter 8. Performing Java analysis 175

Setting options for Java analysis . . . . .	175
Selecting a Java dump data set . . . . .	175
Java fault entry reanalysis . . . . .	176
Displaying the Java information in the interactive report . . . . .	177

## **Chapter 9. The Fault Analyzer report 183**

General report information . . . . .	183
Most significant abend code . . . . .	183
Open file record information . . . . .	183
COBOL suppressed copybooks . . . . .	185
Main report sections . . . . .	185
The prolog section. . . . .	186
The synopsis section . . . . .	186
The summary section. . . . .	186
The event details section . . . . .	188
The system-wide information section . . . . .	189
The abend job information section . . . . .	189
The options section . . . . .	190
The epilog section. . . . .	190
Sample reports . . . . .	191

## **Chapter 10. Using non-ISPF interfaces to access Fault Analyzer history files . 193**

Using the Fault Analyzer plug-in for Eclipse . . . . .	193
Using the Fault Analyzer client for IBM Rational Developer for System z . . . . .	193
Performing interactive reanalysis under CICS . . . . .	194
Using the Fault Analyzer web browser interface . . . . .	194
Viewing all fault entries in a specified history file . . . . .	195
Changing Headings . . . . .	195
Viewing the real-time report of a specific fault entry . . . . .	196
Viewing the last ten accessed history files for a given TSO/ISPF user. . . . .	197

## **Part 2. Fault Analyzer installation and administration . . . . . 199**

### **Chapter 11. Migrating from an earlier version of Fault Analyzer . . . . . 205**

Migrating from V11.1 to V12.1. . . . .	205
Migrating from V10.1 to V11.1. . . . .	205
Migrating from V9.1 to V10.1 . . . . .	205
Migrating from V8.1 to V9.1 . . . . .	206
Migrating from V7.1 to V8.1 . . . . .	206
Migrating from V6.1 to V7.1 . . . . .	207
LPA module compatibility . . . . .	209
Sharing of history files across a sysplex with mixed levels of Fault Analyzer . . . . .	209

### **Chapter 12. Preparing to customize Fault Analyzer . . . . . 213**

Checklist for installing and customizing Fault Analyzer . . . . .	213
Library names after you finish installing . . . . .	216
Storage recommendations . . . . .	218
Exits for invoking Fault Analyzer. . . . .	219

Invocation for non-CICS transaction abends . . . . .	219
Invocation for CICS transaction abends. . . . .	221
SVC dump registration . . . . .	222
Language Environment options required for invocation of Fault Analyzer . . . . .	222
LE options required for non-CICS abends . . . . .	222
LE options required for CICS abends . . . . .	223
Running Fault Analyzer with similar third-party products . . . . .	223
MVS dump data set size. . . . .	224
Application-handled error conditions . . . . .	224

## **Chapter 13. Customizing the operating environment for Fault Analyzer. . . . 225**

Making Fault Analyzer modules available. . . . .	225
Defining program control access to Fault Analyzer programs. . . . .	226
Restricting change of history file settings . . . . .	226
Setting up the message and abend code explanation repository . . . . .	227
Managing recovery fault recording data set access . . . . .	228
SDUMP recovery fault recording data sets. . . . .	229
TDUMP recovery fault recording data sets . . . . .	229
RFR TDUMP XFACILIT example. . . . .	231

## **Chapter 14. Using the Fault Analyzer IDIS subsystem . . . . . 233**

Sysplex-wide subsystem inter-communication . . . . .	234
Caching of history file \$INDEX data . . . . .	234
Starting the IDIS subsystem . . . . .	235
IDIS subsystem storage requirements . . . . .	236
IDIS subsystem requirements for DB2 . . . . .	237
IDIS subsystem requirements for Java . . . . .	237
Stopping the IDIS subsystem . . . . .	238

## **Chapter 15. Modifying your ISPF environment . . . . . 239**

Allocating ISPF data sets . . . . .	239
Making the Fault Analyzer IDISCMDS command table available . . . . .	239
Updating the ISPF selection panel . . . . .	240
Invoking Fault Analyzer using an ISPF 3.4 line command (FA) . . . . .	240
Invoking Fault Analyzer from SDSF (SFA command) . . . . .	241
Invoking the LOOKUP command using cursor selection (LOOKC command) . . . . .	241
Providing ISPF interface defaults for new users . . . . .	242
Providing installation-specific batch reanalysis JCL control statements. . . . .	242

## **Chapter 16. Customize Fault Analyzer by using USERMODs . . . . . 243**

Enabling Fault Analyzer to be invoked . . . . .	243
Installing the MVS change options/suppress dump exit IDIXDCAP (++)IDITABX). . . . .	243
Enabling the Language Environment abnormal termination exit IDIXCEE . . . . .	243

Working with applications that use a non-Language Environment run time . . . . .	244
Identifying the LE run-time library (++)IDILEDSD) . . . . .	244
Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++)IDISPLI/++)IDISPLIA) . . . . .	245
Always invoking Fault Analyzer from PL/I PLIDUMP (++)IDISPDM) . . . . .	245
Eliminating the need for a dump DD statement (++)IDITABD) . . . . .	245
Specifying an alternative parmlib data set for IDICNF00 (++)IDISCNF) . . . . .	246
Changing the default recovery fault recording IEATDUMP data set name (++)IDISRFR) . . . . .	246
Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (++)IDISXCUM) . . . . .	247
Obtaining load modules from CA-Panexec . . . . .	247

## Chapter 17. Setting up history files . . . . . 249

Determining what size history files to allocate . . . . .	249
Allocating a PDS or PDSE for a history file . . . . .	249
AUTO-managed PDSE history files . . . . .	250
Providing the name of a history file to Fault Analyzer . . . . .	250
Setting up views . . . . .	250
Specifying a default column layout . . . . .	252
Specifying an initial fault entry selection criteria . . . . .	252
View considerations if not using XFACILIT resource class . . . . .	253
Managing history files across MVS systems without shared DASD . . . . .	253
Automated implementation . . . . .	254
On demand implementation . . . . .	258
Sharing of history files across a sysplex. . . . .	260
Managing history file fault entry access . . . . .	261
Using the XFACILIT resource class for history file fault entries . . . . .	261

## Chapter 18. Setting and changing default options for the site . . . . . 267

Parmlib member IDICNFxx . . . . .	267
Controlling which jobs are analyzed with Exclude processing . . . . .	269
Fast Exclude options processing . . . . .	270
Controlling report detail. . . . .	270
Limiting the size of minidumps . . . . .	271
Pointing to listings and REXX exec libraries . . . . .	271
Suppressing noncritical SYSLOG messages . . . . .	271
Customizing the Fault Entry List display . . . . .	271
Specifying the cultural environment . . . . .	271

## Chapter 19. Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products . . . . . 273

Updating your build process . . . . .	274
Updating your promotion process . . . . .	275
Preparing your programs . . . . .	275

Enterprise COBOL for z/OS Version 4 programs . . . . .	275
Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs . . . . .	278
COBOL for MVS and VM programs . . . . .	280
VS COBOL II programs . . . . .	282
OS/VS COBOL programs . . . . .	284
Enterprise PL/I Version 3.7 and later programs . . . . .	285
Enterprise PL/I Version 3.5 and Version 3.6 programs . . . . .	288
Enterprise PL/I Version 3.4 and earlier programs . . . . .	291
PL/I for MVS and VM and OS PL/I programs . . . . .	294
z/OS XL C and C++ programs . . . . .	296
Assembler programs . . . . .	300

## Chapter 20. Providing compiler listings or Fault Analyzer side files . . . . . 303

Creating side files using IDILANGX. . . . .	303
IDILANGX parameters . . . . .	305
Side file compatibility with Debug Tool for z/OS . . . . .	306
Including an IDILANGX step in your SCLM translator. . . . .	306
High Level Assembler SCLM example . . . . .	306
COBOL SCLM example . . . . .	307
COBOL Report Writer Precompiler . . . . .	307
Required compiler options for IDILANGX. . . . .	308
TEST option considerations. . . . .	309
DEBUG option considerations. . . . .	310
Naming compiler listings or side files . . . . .	310
Naming CSECTs for Fault Analyzer . . . . .	311
Locating compiler listings or side files . . . . .	311
IDITRACE information . . . . .	313
Compiler listings and side file attributes . . . . .	313
Using the IDIRLOAD DDname for CSECT mapping . . . . .	314
IDILANGP side file formatting utility . . . . .	315
ISPF-packed compiler listings . . . . .	316

## Chapter 21. Customizing the CICS environment . . . . . 319

Configuring Language Environment for CICS to invoke Fault Analyzer . . . . .	320
Defining required programs to CICS . . . . .	320
Adding the required programs to the startup PLT . . . . .	320
Adding the required programs to the shutdown PLT . . . . .	321
Enabling dynamic control of analysis of CICS transaction abends. . . . .	321
Using CFA to FORCEPURGE the currently analyzed task . . . . .	321
SVC dump screening. . . . .	321
Sample definition job. . . . .	322
Controlling CICS transactionabend analysis . . . . .	323
Using CFA from a CICS terminal. . . . .	323
Using CFA from an MVS console. . . . .	325
Ensuring transactionabend analysis is not suppressed by DUMP(NO) . . . . .	326
CICS NoDup(CICSFAST) override assembler exit (IDINDFUE). . . . .	326



Invocation . . . . .	326
Preventing LE from causing the CICS trace to wrap	328
Specifying CICS options through the IDIOPTS	
DDname . . . . .	328
Language Environment abend considerations.	329
Capture of abends running on CICS user key open	
TCBs (L9 TCBs) . . . . .	329
Installing the MVS post-dump exit IDIXTSEL . . . . .	329
Storage requirements . . . . .	329
Maximizing CICS transaction abend analysis	
performance. . . . .	329

## **Chapter 22. Customizing the DB2 environment . . . . . 331**

Binding DB2. . . . .	331
DB2 and Language Environment . . . . .	331
DB2 stored procedures . . . . .	331
Improving Fault Analyzer DB2 performance . . . . .	333

## **Chapter 23. Customizing the IMS environment . . . . . 335**

IMS and Language Environment . . . . .	335
--	-----

## **Chapter 24. Customizing the Fault Analyzer Japanese feature. . . . . 337**

Allocating ISPF data sets . . . . .	337
Setting the national language . . . . .	337

## **Chapter 25. Customizing the Fault Analyzer Korean feature. . . . . 339**

Allocating ISPF data sets . . . . .	339
Setting the national language . . . . .	339

## **Chapter 26. Verifying the customization of Fault Analyzer . . . 341**

Verifying the use of Fault Analyzer with assembler	341
Verifying the use of Fault Analyzer with COBOL	342
Verifying the use of Fault Analyzer with PL/I . . . . .	343
Verifying the use of Fault Analyzer with C . . . . .	344
Verifying the IDIXCEE Language Environment exit	
enablement . . . . .	345
Verifying the IDITABD USERMOD installation . . . . .	345
Verifying the customization of Fault Analyzer	
under CICS . . . . .	345
CICS IVP: 0C1 in program IDIXFA . . . . .	346
CICS IVP: EXEC CICS DUMP	
DUMPCODE(FAD1) . . . . .	346
CICS IVP: EXEC CICS ABEND ABCODE(FLT1)	347
CICS IVP: EXEC CICS ABEND ABCODE(FLT2)	347
Verifying the use of Fault Analyzer with DB2 . . . . .	347
Using a C program . . . . .	347
Using a COBOL program . . . . .	349
Verifying the use of Fault Analyzer through ISPF	351
Verifying the recovery fault recording set-up . . . . .	351

## **Chapter 27. Managing history files (IDIUTIL utility). . . . . 353**

IDIUTIL control statements. . . . .	354
-------------------------------------	-----

FILES control statement . . . . .	354
LISTHF control statement . . . . .	354
DELETE control statement . . . . .	355
SETFAULTPREFIX control statement . . . . .	356
SETMAXFAULTENTRIES control statement . . . . .	357
IMPORT control statement . . . . .	358
EXITS control statement . . . . .	359
Examples. . . . .	360
Example 1. Listing history file entries . . . . .	360
Example 2. Deleting history file entries by date	360
Example 3. Deleting history file entries by	
utilization . . . . .	360
Example 4. Changing history file fault prefix	
characters . . . . .	360
Example 5. Creating a self-maintained history	
file . . . . .	361
Example 6. Importing history file entries . . . . .	361
IDIUTIL batch utility user exit samples. . . . .	362

## **Chapter 28. Providing explanations for application-specific messages . . . . . 363**

Message search order. . . . .	364
Refreshing cached messages . . . . .	364

## **Chapter 29. Maintaining Fault Analyzer . . . . . 365**

## **Chapter 30. Disabling Fault Analyzer 367**

Temporarily deinstalling Fault Analyzer . . . . .	367
Turning off Fault Analyzer using the IFAPRDxx	
parmlib member . . . . .	367
Turning off Fault Analyzer with a JCL switch	
(IDIOFF) . . . . .	368
Turning off Fault Analyzer using an environment	
variable (_IDI_OFF) . . . . .	368

## **Chapter 31. Customizing Fault Analyzer by using user exits. . . . . 369**

Invocation parameters . . . . .	372
Global environment data area (ENV) . . . . .	372
User exit type specific data area . . . . .	372
Supported exit programming languages . . . . .	372
Data area version checking . . . . .	373
Diagnostic tracing . . . . .	373
Tracing user exit parameter list values . . . . .	373
Tracing REXX EXECs. . . . .	377
Descriptions of fault analysis user exit types . . . . .	377
Analysis Control user exit . . . . .	377
Analysis Control user exit (Dump registration)	380
Compiler Listing Read user exit . . . . .	382
Message and Abend Code Explanation user exit	386
Formatting user exit . . . . .	390
End Processing user exit. . . . .	402
End Processing user exit (Fault entry refresh)	404
Notification user exit . . . . .	405
Notification user exit (Dump registration) . . . . .	411
Descriptions of IDIUTIL batch utility user exit	
types . . . . .	412
IDIUTIL Import user exit . . . . .	412

IDIUTIL Delete user exit . . . . .	413
IDIUTIL ListHF user exit . . . . .	414
User exit REXX commands . . . . .	418
Evaluate command . . . . .	418
IDIALLOC command. . . . .	420
IDIDDDTEST command . . . . .	424
IDIDSECTdsn command. . . . .	424
IDIEventInfo command . . . . .	425
IDIFREE command . . . . .	426
IDIModQry command . . . . .	426
IDIRegisterFaultEntry command . . . . .	427
IDIWRITE command . . . . .	428
IDIWTO command . . . . .	429
List command . . . . .	430
Note command. . . . .	432
Formatting tags . . . . .	433
ADDR (address) . . . . .	436
AREA (area). . . . .	436
DD (definition description). . . . .	437
DATA (data). . . . .	437
DL (definition list). . . . .	437
DT (definition term) . . . . .	438
DUMP (EBCDIC dump). . . . .	439
DUMPA (ASCII dump) . . . . .	439
HP (highlighted phrase). . . . .	440
L (line) . . . . .	440
LI (list item). . . . .	441
NOTEL (note list) . . . . .	441
P (paragraph) . . . . .	441
TH (table heading) . . . . .	442
U (underline) . . . . .	442
UL (unordered list) . . . . .	443

## Chapter 32. Installing non-ISPF interfaces to access Fault Analyzer history files . . . . . 445

Installing the Fault Analyzer plug-in for Eclipse . . . . .	445
Customize the IBM Problem Determination Tools for z/OS Common Component Common Server . . . . .	445
Download and install CICS Explorer . . . . .	446
Download Fault Analyzer plug-in . . . . .	446
Install Fault Analyzer plug-in into CICS Explorer . . . . .	446
Installing the Fault Analyzer client for IBM Rational Developer for System z . . . . .	446
Enabling interactive reanalysis under CICS . . . . .	447
Making the required CICS resource definitions . . . . .	447
Making the required CICS JCL changes. . . . .	447
Installing the Fault Analyzer web browser interface . . . . .	448
Running the sample job to create a program object in a z/OS Unix System Services file. . . . .	448
Making the required updates to the HTTP server configuration files . . . . .	448

## Part 3. Fault Analyzer reference information . . . . . 449

### Chapter 33. Options . . . . . 451

Where to specify options . . . . .	452
Parmlib member IDICNF00. . . . .	453
User-options module IDICNFUM. . . . .	453
Configuration-options module IDIOPTLM. . . . .	454
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 . . . . .	458
DeferredReport. . . . .	463
Detail . . . . .	465
DumpDSN . . . . .	466
DumpRegistrationExits . . . . .	466
ErrorHandler . . . . .	467
Exclude/Include . . . . .	468
Exits . . . . .	473
FaultID . . . . .	475
GenerateSavedReport. . . . .	476
HistCols . . . . .	477
Include . . . . .	478
InteractiveExitPromptSeconds . . . . .	478
Language. . . . .	479
Locale . . . . .	480
LoopProtection . . . . .	480
MaxMinidumpPages . . . . .	481
NoDup . . . . .	482
PermitLangx. . . . .	490
PreferredFormattingWidth . . . . .	491
PrintInactiveCOBOL . . . . .	492
Quiet . . . . .	492
RDZClient . . . . .	493
RefreshExits . . . . .	493
RetainCICSDump . . . . .	495
RetainDump. . . . .	495
Source. . . . .	496
SpinIDIREPRT . . . . .	497
StoragePrintLimit . . . . .	497
StorageRange . . . . .	498
SystemWidePreferred. . . . .	499
UseIDISTime . . . . .	501

### Chapter 34. Data areas . . . . . 503

Non-REXX user exit buffered data format . . . . .	503
Data area descriptions . . . . .	505
CTL - Analysis Control user exit parameter list . . . . .	505
ENV - Common exit environment information . . . . .	512
EPC - End Processing user exit parameter list . . . . .	519
LST - Compiler Listing Read user exit parameter list . . . . .	520
NFY - Notification user exit parameter list. . . . .	522
UFM - Formatting user exit parameter list. . . . .	523
UTL - IDIUTIL Batch Utility user exit parameter list . . . . .	529
XPL - Message and Abend Code Explanation user exit parameter list . . . . .	530

### Chapter 35. Return codes . . . . . 533



Batch reanalysis (IDIDA) . . . . .	533
IDIUTIL batch utility . . . . .	533
IDILANGX . . . . .	533

## **Chapter 36. Messages. . . . . 535**

Fault Analyzer messages . . . . .	535
IDILANGX messages. . . . .	556

## **Appendix A. Support resources and problem solving information . . . . . 557**

Searching knowledge bases. . . . .	557
Searching the information center . . . . .	557
Searching product support documents . . . . .	557
Getting fixes. . . . .	559
Subscribing to support updates . . . . .	559
RSS feeds and social media subscriptions . . . . .	559
My Notifications . . . . .	559
Contacting IBM Support. . . . .	560
Define the problem and determine the severity of the problem . . . . .	561

Gather diagnostic information. . . . .	562
Submit the problem to IBM Support. . . . .	562

## **Appendix B. Sample customized ISPF interface front-end . . . . . 565**

Fault History File or VIEW name. . . . .	567
MATCH on program name. . . . .	567
Installing the sample application . . . . .	567
How the sample works . . . . .	567

## **Notices . . . . . 569**

Programming interface information . . . . .	570
Trademarks . . . . .	571

## **Glossary . . . . . 573**

## **Index . . . . . 575**



---

## 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 may also wish 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 199 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 issued by Fault Analyzer.

Appendix A, “Support resources and problem solving information,” on page 557 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 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 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

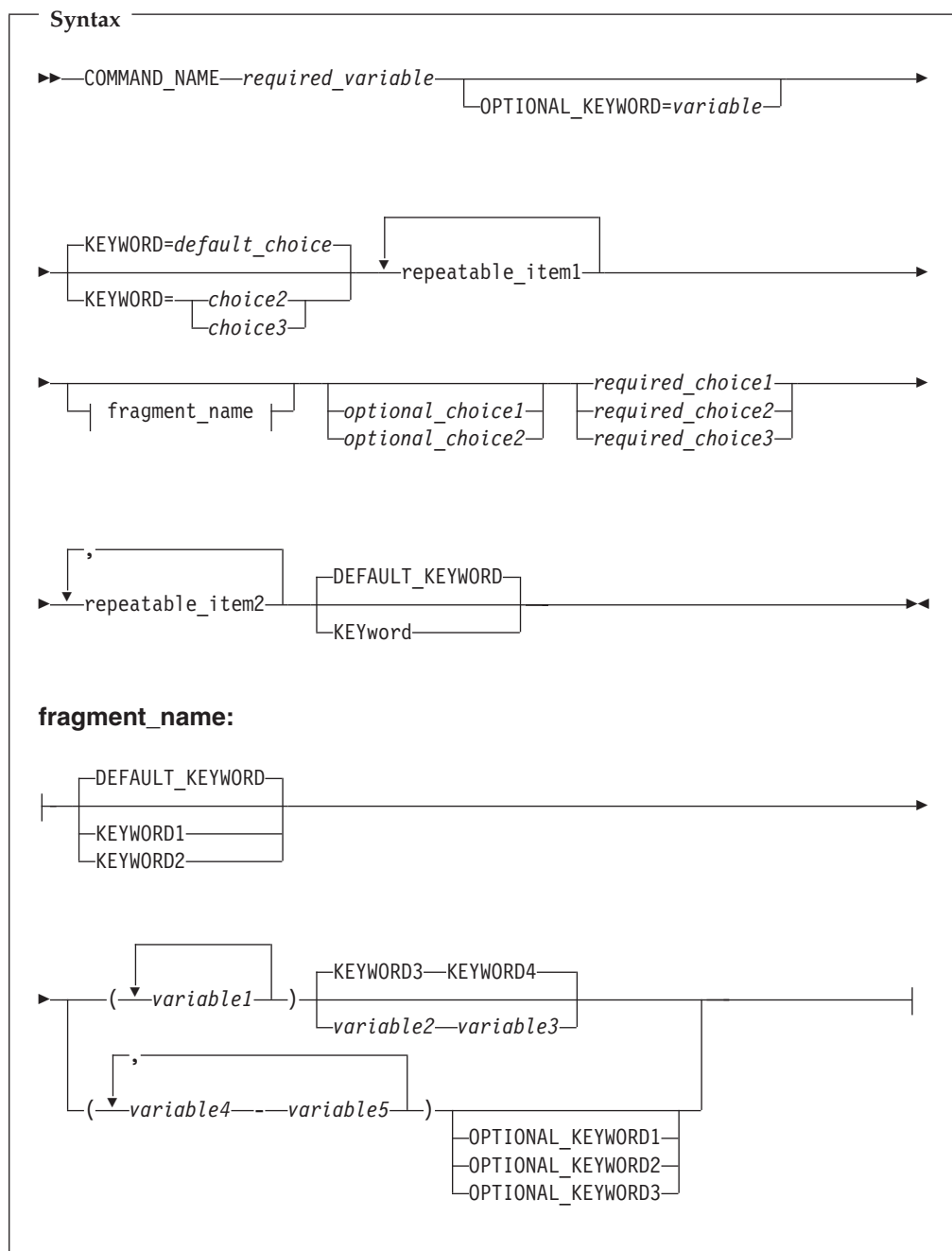


Figure 1. Sample syntax diagram

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.

## How to read the syntax diagrams

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—

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


——  
——

In this example, you could type "KEY", "KEYW", "KEYWO", "KEYWOR" 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\_variable*—




### Required items

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

—COMMAND\_NAME—*required\_variable*—

### Optional items


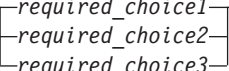

Optional items appear below the main path.

——

### 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.

## How to read the syntax diagrams



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



### Repeatable items

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

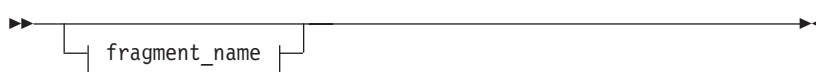


If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.



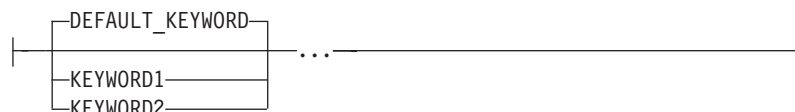
### Fragments

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



⋮

**fragment\_name:**





---

## 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  
SC19-3671-00
- The topic and page number 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 will only use 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 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

---

### IBM Fault Analyzer for z/OS Version 9 Release 1

The following describes the changes for this version.

#### SC19-2538-00: September 2008

This edition of the book provides information applicable to Fault Analyzer Version 9 Release 1.

Changes in this edition include:

- 64-bit support has been added to the following:
  - The Formatting user exit formatting tags that are used to display storage locations:
    - ADDR
    - DUMP
    - DUMPA
  - For details, see “Formatting tags” on page 433.
  - The SHOW command—for details, see “SHOW” on page 69.
  - The Dump Storage display—for details, see “Displaying storage locations” on page 138.
- Additional line commands have been added to the Fault Entry List display to facilitate fault entry Copy, Move, or XMIT—for details, see “Applying an action to a particular fault” on page 49.
- An additional CICS IVP has been provided using LE-compliant assembler—for details, see “CICS IVP: EXEC CICS ABEND ABCODE(FLT2)” on page 347.
- The LOOKUP command has also been made available for use outside of Fault Analyzer—for details, see “LOOKUP” on page 66.
- Additional ways to invoke Fault Analyzer and the LOOKUP command have been provided—for details, see Chapter 15, “Modifying your ISPF environment,” on page 239.
- An example of how one can define security access to a common history file using the XFACILIT resource class has been provided—for details, see “XFACILIT implementation example 1” on page 263.
- Information about sharing of history file data sets across a sysplex has been updated—for details, see “Sharing of history files across a sysplex” on page 260.
- A new section has been added, which can help you answer questions and solve problems—for details, see Appendix A, “Support resources and problem solving information,” on page 557.

Following the product install, this version of the book is available in both PDF and BookManager format in the data sets IDL.SIDIDOC1(IDIUGPDF) and IDL.SIDIBOOK(IDIUGBOO) respectively.

#### SC19-2538-01: APAR PK74751 edition (January 2009)

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

- A new USERMOD has been provided for non-LE PL/I V2R3, which might be suitable for IMS or similar environments, where an abend following the

## SC19-2538-01: APAR PK74751 edition (January 2009)

IDISNAP call is desired—for details, see “Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++IDISPLI/++IDISPLIA)” on page 245.

- A new USERMOD that can be used to change the behavior of Fault Analyzer when invoked via the CICS XDUREQ global user exit has been provided—for details, see “Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (++IDISXCUM)” on page 247.
- An additional example of how one can define security access to a common history file using the XFACILIT resource class has been provided—for details, see “XFACILIT implementation example 2: Using global access table” on page 263.
- Additional information about the IDIJLIB path used by the IDIJ subsystem has been provided.
- A new option, XCFGRPSUFFIX, has been provided for the IDIS subsystem—for details, see “Starting the IDIS subsystem” on page 235.
- The ability to test the recovery fault recording set-up has been provided—for details, see “Verifying the recovery fault recording set-up” on page 351.
- Information about compatibility if sharing history files between sysplex images with a mix of Fault Analyzer versions or maintenance levels installed has been added. For details, see “Sharing of history files across a sysplex with mixed levels of Fault Analyzer” on page 209.
- The following messages have been modified:
  - IDI0106E
  - IDI0123S

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

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PK74751 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-2538-02: APAR PK79442 edition (March 2009)

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

- Information about Language Environment options required to capture Java application abends has been provided—for details, see “LE options required to capture Java application abends” on page 223.
- Improvements have been made to the Compiler Listing Not Found display, and subsequent validation of provided compiler listings or side files—for details, see “Prompting for compiler listing or side file” on page 156.
- A sample Formatting user exit has been provided for batch registration of MVS dumps in a history file—for details, see “Example 4 (Batch MVS dump registration)” on page 401.
- The following messages have been added:
  - IDI0145I
  - IDI0146I

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

- The following messages have been modified:
  - IDI0053I
  - IDI0112W

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

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PK79442 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

### **SC19-2538-03: APAR PK89333 edition (July 2009)**

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

- Additional information about the use of the IDIOPTS DDname with CICS has been provided in “Specifying CICS options through the IDIOPTS DDname” on page 328.
- A new LE-compliant assembler IVP has been provided as an alternative to the COBOL and PL/I IVPs for verification of the IDIXCEE Language Environment invocation exit enablement. For details, see “Verifying the IDIXCEE Language Environment exit enablement” on page 345.
- Support for sorting and filtering of CICS system dump analysis table displays has been added—for details, see “Sorting and matching table displays” on page 169.
- Two changes to recovery fault recording (RFR) processing have been introduced:
  - To eliminate RFR dump data set management problems, Fault Analyzer no longer replaces the IDIRFRDS CSECT name high-level qualifier by the user ID associated with the abending job, if failing to allocate the IDIRFRDS CSECT name.
  - To improve performance, SDUMP support has been added as the preferred RFR dump type, the use of which is controlled by the IDI\_SDUMP\_ACCESS XFACILIT resource class.

With the recommended UACC(NONE) access to the normal IDIRFRDS CSECT data set name profile, both SDUMP and IEATDUMP RFR dump types can be controlled using XFACILIT access.

For details, see “Recovery fault recording” on page 28 and “Managing recovery fault recording data set access” on page 228.

- Information about the WTO routing and descriptor codes used by Fault Analyzer has been provided on page 12.
- Sample report displays updated to show Fault Analyzer UNICODE support: Figure 46 on page 105 and Figure 71 on page 131.
- Sample report display updated to show COBOL PERFORM traceback: Figure 48 on page 108.
- The following messages have been added:
  - IDI0147W
  - IDI0148I
  - IDI0149W
  - IDI0150W
  - IDI0151W

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

- The following messages have been modified:
  - IDI0011S
  - IDI0127W
  - IDI0135E

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

## SC19-2538-03: APAR PK89333 edition (July 2009)

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PK89333 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-2538-04: APAR PK92822 edition (September 2009)

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

- The section “Managing recovery fault recording data set access” on page 228 has been extensively revised.
- The following messages have been added:
  - IDI0152I
  - IDI0153I

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

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PK92822 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

---

## IBM Fault Analyzer for z/OS Version 10 Release 1

The following describes the changes for this version.

### SC19-2868-00: November 2009

This edition of the book provides information applicable to Fault Analyzer Version 10 Release 1.

Changes in this edition include:

- A new section has been added, which describes the minimal steps required to prepare your programs for use with the IBM Problem Determination Tools products—for details, see Chapter 19, “Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products,” on page 273.
- Sorting and matching improvements have been made to the Fault Entry List display. For details, see “Sorting and matching fault entries” on page 45.
- The ability to allocate a new history file as the target for a Fault Entry List display Move or Copy line command, or simply by using the Fault Entry List display action-bar File pull-down menu, has been added. For details, see “New history file allocation” on page 53.
- A user notes option has been added to the initial interactive reanalysis report display when the fault entry contains one or more user notes. For details, see “Primary option: User Notes” on page 129.
- Support for an configuration-options load module, as an alternative to a number of SMP/E USERMODs, has been provided. For details, see “Configuration-options module IDIOPTLM” on page 454.
- Improvements have been made to the Java support, which includes the use of the IDIS subsystem instead of the earlier IDIJ subsystem. For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 233 and Chapter 8, “Performing Java analysis,” on page 175.
- Information about migrating from Fault Analyzer V9.1 to V10.1 has been provided in “Migrating from V9.1 to V10.1” on page 205.
- The following messages have been added:
  - IDI0154W



- IDI0155W

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

Following the product install, this version of the book is available in both PDF and BookManager format in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-2868-01: APAR PM00092 edition (December 2009)

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

- A softcopy sample job to create an IDIOPTLM configuration-options load module has been provided. For details, see “Configuration-options module IDIOPTLM” on page 454.
- The ability to specify a history file name using wildcards, and then select a history file from a list of matches, has been provided for the Fault Entry List display. For details, see “Changing the history file or view displayed” on page 35.
- An example of how to provide history file XFACILIT access with ACF2 has been provided. For details, see “XFACILIT implementation example 3: Using ACF2” on page 263.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM00092 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-2868-02: APAR PM08932 edition (April 2010)

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

- Additional CICS system dump analysis domains have been added. For details, see Figure 100 on page 169.
- The Debug Tool for z/OS EQAUEDAT exit is now called for all program languages when searching for a matching compiler listing or side file. For details, see “Locating compiler listings or side files” on page 311.
- Changes to user exit data area fields:
  - The following user exit data area fields have been added:
    - ENV.POF\_LOADED\_FROM
    - ENV.EXEC\_LOADED\_FROM
    - ENV.DUP\_DATE
    - ENV.DUP\_TIME
    - ENV.GROUP\_ID
    - ENV.INVOCATION\_ABEND\_CODE
    - ENV.MINIDUMP\_PAGES
  - The following user exit data area fields have been deprecated:
    - EPC.MINIDUMP\_PAGES (use ENV.MINIDUMP\_PAGES instead)

For details, see Chapter 34, “Data areas,” on page 503.

- Changes to messages:
  - The following messages have been added:
    - IDI0156W
  - The following messages have been modified:
    - IDI0133W

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

## **SC19-2868-02: APAR PM08932 edition (April 2010)**

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM08932 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## **SC19-2868-03: APAR PM11905 edition (May 2010)**

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

- A Fault Analyzer Eclipse plug-in for CICS Explorer has been provided—for details, see “Using the Fault Analyzer plug-in for Eclipse” on page 193.
- Descriptions of all non-ISPF interfaces to history files have been moved to two new chapters: Chapter 10, “Using non-ISPF interfaces to access Fault Analyzer history files,” on page 193 and Chapter 32, “Installing non-ISPF interfaces to access Fault Analyzer history files,” on page 445.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM11905 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## **SC19-2868-04: APAR PM14420 edition (August 2010)**

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

- A new option, CICSTraceMax, has been added. This option is only applicable to CICS system dump analysis. For details, see “CICSTraceMax” on page 456.
- Changes to messages:
  - The following messages have been added:
    - IDI0157I
  - The following messages have been modified:
    - IDI0033E
    - IDI0036E
    - IDI0043W
    - IDI0123S
    - IDI0151W
  - The following messages have been removed:
    - IDI0145I

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

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM14420 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## **SC19-2868-05: APAR PM21096 edition (October 2010)**

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

- Improvements have been made to PDSE history file AUTO-space management by allowing a minimum number of fault entries to exist before deleting old fault entries. For details, see “AUTO-managed PDSE history files” on page 250 and “SETMAXFAULTENTRIES control statement” on page 357.
- Changes to Java dump analysis processing have been made. For details, see Chapter 8, “Performing Java analysis,” on page 175.

- Information about how to invoke Fault Analyzer from Java has been provided in “Invoking Fault Analyzer from Java dump events” on page 24 and “Invoking Fault Analyzer from Java catch block” on page 22.
- Information about recommended compiler options used for Enterprise PL/I version 3.6 and 3.7 programs has been added. For details, see Chapter 19, “Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products,” on page 273.
- Information about DB2 V10 support has been added. For details, see “Supported application environments” on page 9.
- The following messages have been removed:
  - IDI0148I
- If sharing history files across a sysplex, it is no longer necessary to ensure that major name IDIINDEX is shared in the same way as SPFEDIT, since the use of major name IDIINDEX has been removed.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM21096 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## **SC19-2868-06: APAR PM27330 edition (February 2011)**

This version of the book contains minor clarifications and corrections.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM27330 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

---

## **IBM Fault Analyzer for z/OS Version 11 Release 1**

The following describes the changes for this version.

### **SC19-3131-00: November 2010**

This edition of the book provides information applicable to Fault Analyzer Version 11 Release 1.

Changes in this edition include:

- The ability to format a CICS auxiliary trace has been provided. For details, see Chapter 7, “Formatting a CICS auxiliary trace data set,” on page 173.
- The following messages have been added:
  - IDI0158W

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

- Information about migrating from Fault Analyzer V10.1 to V11.1 has been provided in “Migrating from V10.1 to V11.1” on page 205.

Following the product install, this version of the book is available in both PDF and BookManager format in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

### **SC19-3131-01: APAR PM28208 Edition (February 2011)**

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

- The New History File Allocation display has been simplified. For details, see “New history file allocation” on page 53.

## SC19-3131-01: APAR PM28208 Edition (February 2011)

- Information about APF-authorization of the IDI.SIDIAUT2 SMP/E target data set has been provided in “Making Fault Analyzer modules available” on page 225.
- Information about Fault Analyzer recovery fault recording dump titles used has been provided in “IEATDUMP dump title” on page 30.
- Point-and-shoot enabled abend codes have been provided on the Fault Entry List display. For details, see “The Fault Entry List display” on page 32.

**Note:** User abend codes will only be point-and-shoot enabled on this display if the fault entry was created at PTF level UK65276, or later.

- Changes to user exit data area fields:
  - The following user exit data area fields have been added:
    - UFM.FPREG0 through UFM.FPREG15
  - The following user exit data area fields have been deprecated:
    - UFM.NUM\_FPREGS
    - UFM.FPREG\_DATA\_ADDRESS
  - The following user exit data area fields have been changed:
    - XPL.ABEND\_CODE (Offset and length)

For details, see Chapter 34, “Data areas,” on page 503.

- The following messages have been added:
  - IDI0159I

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

- Improvements have been made to the Fault Analyzer plug-in for Eclipse. For details, see the readme.txt file included in IDI.SIDIDOC2(IDIGUIP). For information about extracting the files contained in this member, and installing the updated plug-in, see “Installing the Fault Analyzer plug-in for Eclipse” on page 445.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM28208 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-3131-02: APAR PM34295 Edition (June 2011)

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

- A new option, GenerateSavedReport, has been provided. For details, see “GenerateSavedReport” on page 476.
- A new interactive reanalysis option, “Prompt for missing side files”, has been provided. For details, see “Migrating from V10.1 to V11.1” on page 205 and “Interactive reanalysis options” on page 97.
- Information about MDBG side files has been provided. For details, see Chapter 20, “Providing compiler listings or Fault Analyzer side files,” on page 303.
- The following messages have been added:
  - IDI0160I
  - IDI0161W
  - IDI0162I

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

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM34295 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-3131-03: APAR PM44693 Edition (September 2011)

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

- Additional information about the RACF requirement to include real XFACILIT profiles to back any global access list XFACILIT profiles has been provided in “XFACILIT implementation example 2: Using global access table” on page 263.
- The ability to collapse or expand COBOL level 88 items from the Associated Storage Areas display has been provided. For details, see “Collapsing level 88 items” on page 132.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM44693 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-3131-04: APAR PM47919 Edition (December 2011)

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

- Registration of a fault entry for an MVS dump analyzed in batch can now be performed using the GenerateSavedReport option, as well as the IDIRegisterFaultEntry user exit REXX command. For details, see “GenerateSavedReport” on page 476.
- A new user exit data area field has been added: UFM.FPCR. For details, see “UFM - Formatting user exit parameter list” on page 523.
- Support for options-specification through environment variables \_IDI\_OPTS and \_IDI\_OPTSFILE has been added. For details, see Chapter 33, “Options,” on page 451.
- An enhancement has been made to the CICS CFA transaction to show default CICS options in effect. For details, see “Using CFA from a CICS terminal” on page 323.
- Support for a new DDname, IDIRLOAD, has been added. For details, see “Using the IDIRLOAD DDname for CSECT mapping” on page 314.
- The section “Invoking Fault Analyzer from Java catch block” on page 22 has been revised.
- The following messages have been added:
  - IDI0164IFor details, see Chapter 36, “Messages,” on page 535.
- The following messages have been modified:
  - IDI0029WFor details, see Chapter 36, “Messages,” on page 535.
- The following messages have been removed:
  - IDI0060I

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM47919 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

## SC19-3131-05: APAR PM54422 Edition (February 2012)

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

- PDSE history file utilization has been added to the Fault History File Properties display. For details, see “History file properties” on page 51.
- An additional option has been provided to prevent unwanted SLIP dumps being taken due to expected and handled S0C4 abends in Fault Analyzer. For details, see “Using SLIP,COMP=0C4 with Fault Analyzer” on page 30.

This edition of the book is provided in both PDF and BookManager format with the PTF for APAR PM54422 in the data sets IDI.SIDIDOC1(IDIUGPDF) and IDI.SIDIBOOK(IDIUGBOO) respectively.

---

## **IBM Fault Analyzer for z/OS Version 12 Release 1**

The following describes the changes for this version.

### **SC19-3671-00: May 2012**

This edition of the book provides information applicable to Fault Analyzer Version 12 Release 1.

Changes in this edition include:

- The ability to change history file settings using the Fault Analyzer ISPF interface has been provided. For details, see “Change fault history file settings” on page 54.
- The ability to control the change of history file settings has been provided. For details, see “Restricting change of history file settings” on page 226.
- A new option, AdditionalIDIOffDD, has been added. For details, see “AdditionalIDIOffDD” on page 455.
- Added information about optional XFACILIT user exit, IDIXFXIT. For details, see “Using the IDIXFXIT user exit” on page 264.
- Information about migrating from Fault Analyzer V11.1 to V12.1 has been provided in “Migrating from V11.1 to V12.1” on page 205.
- Changed the LINKLIST requirement for data set IDI.SIDIAUT2. For details, see “Making Fault Analyzer modules available” on page 225.
- Updated the Fault Analyzer plug-in for Eclipse feature installation information due to the replacement of the Java-based Fault Analyzer server by the IBM Problem Determination Tools for z/OS Common Component Common Server. For details, see “Installing the Fault Analyzer plug-in for Eclipse” on page 445.
- Removed already deprecated user exit data area fields:
  - 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)
- Removed IDILANGX return code and message explanations since these now documented in the *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.
- Removed references to the Fault Analyzer User’s Guide and Reference BookManager softcopy version, since this is no longer shipped with Fault Analyzer.
- Removed references to the IDIBOOKS and IDIBxxx suboptions of the DataSets option.
- Removed references to the IDI.SIDIBOOK, IDI.SIDIBJPN and IDI.SIDIBKOR data set names.



## **IBM Fault Analyzer for z/OS Version 12 Release 1**

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

<b>Chapter 1. Introduction</b> . . . . .	3	Available columns . . . . .	42
The analysis engine . . . . .	3	Sorting and matching fault entries . . . . .	45
The analysis process . . . . .	3	Additional ways to match and select faults . . . . .	46
Real-time abend analysis . . . . .	3	Cursor-selecting a matching value . . . . .	46
Real-time SNAP analysis . . . . .	5	Over-typing existing values . . . . .	48
Fault reanalysis . . . . .	6	Using the MATCH command . . . . .	48
Batch reanalysis . . . . .	7	Applying an action to a particular fault . . . . .	49
Interactive reanalysis . . . . .	7	History file properties . . . . .	51
Fault history files . . . . .	8	New history file allocation . . . . .	53
Special members in the history file data set . . . . .	9	Change fault history file settings . . . . .	54
Supported application environments . . . . .	9	Resetting history file access information . . . . .	56
Binder-related dependencies . . . . .	10	Refreshing fault entry information . . . . .	56
Setting up existing programs for fault analysis . . . . .	10	Implicit refresh . . . . .	56
Additional region size required . . . . .	11	Updates pending . . . . .	57
Compiler listing or side file selection criteria . . . . .	11	Fault entry expiration control . . . . .	58
Special processing of Language Environment		Action-bar pull-down menus . . . . .	58
CEEWUCHA abends . . . . .	11	Commands . . . . .	61
WTO routing and descriptor codes used by Fault		COLS . . . . .	61
Analyzer . . . . .	12	COPY . . . . .	62
		DISASM . . . . .	62
<b>Chapter 2. Real-time analysis.</b> . . . . .	13	DSECT . . . . .	62
Dump suppression . . . . .	13	DUPS . . . . .	63
Fault Analyzer and CICS global user exits . . . . .	14	EXEC . . . . .	63
Fault history file selection . . . . .	14	FIND . . . . .	63
Controlling the real-time analysis with options . . . . .	15	FIND command differences between display	
Pointing to listings with JCL DD statements . . . . .	16	types . . . . .	65
The real-time analysis report . . . . .	16	INFO . . . . .	65
Combining Fault Analyzer real-time reports . . . . .	17	LOOKUP . . . . .	66
Controlling the SYSOUT class of real-time reports . . . . .	17	MATCH . . . . .	66
Suppressing real-time reports . . . . .	17	NEXT . . . . .	67
The SYSLOG summary . . . . .	18	NOTELIST . . . . .	67
Using the program SNAP interface (IDISNAP) . . . . .	18	PREV . . . . .	67
IDISNAP invocation . . . . .	18	QUIT . . . . .	68
Entry specifications . . . . .	18	REFRESH . . . . .	68
Return specifications . . . . .	20	RESET . . . . .	68
Example 1 (COBOL) . . . . .	20	RPTFIND . . . . .	68
Example 2 (PL/I) . . . . .	21	RUNCHAIN . . . . .	69
Example 3 (Assembler) . . . . .	22	SHOW . . . . .	69
Invoking Fault Analyzer from Java catch block . . . . .	22	STCK . . . . .	70
Invoking Fault Analyzer from Java dump events . . . . .	24	VIEWS . . . . .	70
Dump registration processing . . . . .	24	Viewing a saved report . . . . .	70
Real-time exclusion processing . . . . .	25	Adding or removing blank lines . . . . .	72
Duplicate fault processing . . . . .	28	Adding or removing help text . . . . .	72
Recovery fault recording . . . . .	28	Setting preferred formatting width . . . . .	72
RFR dump titles . . . . .	30	Displaying user-selected message or abend code	
IEATDUMP dump title . . . . .	30	explanations . . . . .	73
IEATDUMP dump title . . . . .	30	Copying interactive displays to a file . . . . .	75
Using SLIP,COMP=0C4 with Fault Analyzer . . . . .	30	Displaying Fault Analyzer copyright and general	
		usage information . . . . .	76
<b>Chapter 3. The Fault Analyzer ISPF interface</b> . . . . .	31	Deleting history file entries . . . . .	76
Invoking the interface . . . . .	32	Changing deletion options through the Options	
ISPF split screen support . . . . .	32	menu . . . . .	77
The Fault Entry List display . . . . .	32	Deleting when the confirmation display is shown . . . . .	77
Using views . . . . .	35	Deleting when the confirmation display is not	
Changing the history file or view displayed . . . . .	35	shown . . . . .	78
Fault entry list column configuration . . . . .	41	Bulk deletions . . . . .	78

Viewing fault entry information . . . . .	78
Viewing the fault entry duplicate history . . . . .	83
Copying history file entries . . . . .	86
Moving history file entries . . . . .	87
Transmitting history file entries. . . . .	87
Security considerations . . . . .	88

<b>Chapter 4. Performing batch reanalysis . . . . .</b>	<b>89</b>
Batch reanalysis options . . . . .	89
Initiating batch reanalysis. . . . .	94
Data sets used for batch reanalysis . . . . .	94
Creating your own batch reanalysis job . . . . .	95

<b>Chapter 5. Performing interactive reanalysis . . . . .</b>	<b>97</b>
Interactive reanalysis options . . . . .	97
Initiating interactive reanalysis . . . . .	101
Status pop-up display . . . . .	101
General information about the interactive report . . . . .	102
Exit from the interactive report . . . . .	104
Primary option: Synopsis . . . . .	105
Primary option: Event Summary . . . . .	105
Detailed Event Information. . . . .	107
Primary option: Open Files . . . . .	110
Primary option: CICS Information . . . . .	112
Last CICS 3270 Screen Buffer . . . . .	113
Last CICS 3270 Screen Buffer Hex . . . . .	113
Summarized CICS Trace . . . . .	114
CICS Trace Formatting . . . . .	115
CICS Levels, Commareas, and Channels . . . . .	117
Primary option: Messages . . . . .	120
Primary option: DB2 Information. . . . .	120
Primary option: IMS Information. . . . .	123
Primary option: Storage Areas. . . . .	126
Primary option: Java Information. . . . .	127
Primary option: WebSphere Information . . . . .	127
Primary option: Language Environment Heap Analysis . . . . .	127
Primary option: MTRACE Records . . . . .	128
Primary option: Abend Job Information . . . . .	129
Primary option: User Notes . . . . .	129
Primary option: Fault Analyzer Options . . . . .	129
Displaying associated storage areas . . . . .	130
Hiding the hex-value column . . . . .	131
Collapsing level 88 items . . . . .	132
Expanding messages and abend codes . . . . .	134
Displaying source code . . . . .	135
Displaying storage locations . . . . .	138
Creating and managing user notes . . . . .	140
Mapping storage areas using DSECT information . . . . .	145
IDIDSECT concatenation . . . . .	147
Indexing your DSECT data sets (\$DINDEX member) . . . . .	147
DSECT indexing utility (IDIPDSCU). . . . .	148
Displaying chained data areas. . . . .	148
Disassembling object code . . . . .	150
Converting STORE CLOCK values . . . . .	152
User-specific report formatting . . . . .	154
Prompting for compiler listing or side file. . . . .	156
Controlling prompting . . . . .	160
Data sets used for interactive reanalysis . . . . .	160
Refresh processing. . . . .	161

<b>Chapter 6. Performing CICS system abend dump analysis . . . . .</b>	<b>163</b>
Setting options for CICS system abend analysis . . . . .	163
User exits . . . . .	163
Selecting a CICS dump data set . . . . .	163
Selecting an address space to analyze . . . . .	165
Displaying the CICS system abend interactive report . . . . .	166
Fastpath navigation . . . . .	167
Option 1: Synopsis . . . . .	167
Option 2: Abend Job Information. . . . .	167
Option 3: CICS System Information . . . . .	168
Sorting and matching table displays. . . . .	169
Option 4: Options in Effect . . . . .	170
Creating a history file entry . . . . .	171

<b>Chapter 7. Formatting a CICS auxiliary trace data set . . . . .</b>	<b>173</b>
Selecting a CICS auxiliary trace data set . . . . .	173
Specifying CICS Trace Selection Parameters . . . . .	174

<b>Chapter 8. Performing Java analysis. . . . .</b>	<b>175</b>
Setting options for Java analysis . . . . .	175
Selecting a Java dump data set . . . . .	175
Java fault entry reanalysis . . . . .	176
Displaying the Java information in the interactive report . . . . .	177

<b>Chapter 9. The Fault Analyzer report. . . . .</b>	<b>183</b>
General report information . . . . .	183
Most significant abend code . . . . .	183
Open file record information . . . . .	183
Spanned record interpretation . . . . .	183
COBOL suppressed copybooks . . . . .	185
Main report sections . . . . .	185
The prolog section. . . . .	186
The synopsis section. . . . .	186
The summary section. . . . .	186
Maximum call depth . . . . .	187
The event details section . . . . .	188
The system-wide information section . . . . .	189
The abend job information section . . . . .	189
The options section . . . . .	190
The epilog section. . . . .	190
Sample reports . . . . .	191

<b>Chapter 10. Using non-ISPF interfaces to access Fault Analyzer history files . . . . .</b>	<b>193</b>
Using the Fault Analyzer plug-in for Eclipse . . . . .	193
Using the Fault Analyzer client for IBM Rational Developer for System z . . . . .	193
Performing interactive reanalysis under CICS . . . . .	194
Using the Fault Analyzer web browser interface . . . . .	194
Viewing all fault entries in a specified history file . . . . .	195
Changing Headings . . . . .	195
Viewing the real-time report of a specific fault entry . . . . .	196
Viewing the last ten accessed history files for a given TSO/ISPF user. . . . .	197

---

## 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 219 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.

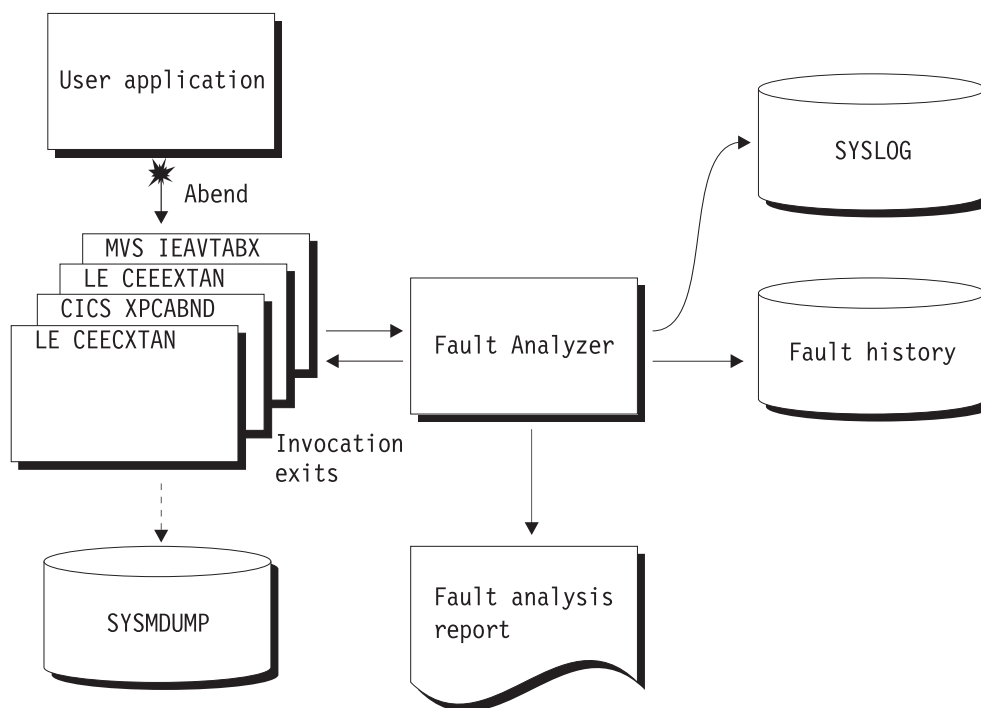


Figure 2. Real-time abend analysis

For an example of a real-time fault analysis report, see “Sample reports” on page 191.

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 481)
- 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 402).

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 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 11 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 will be adequate, and you will 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

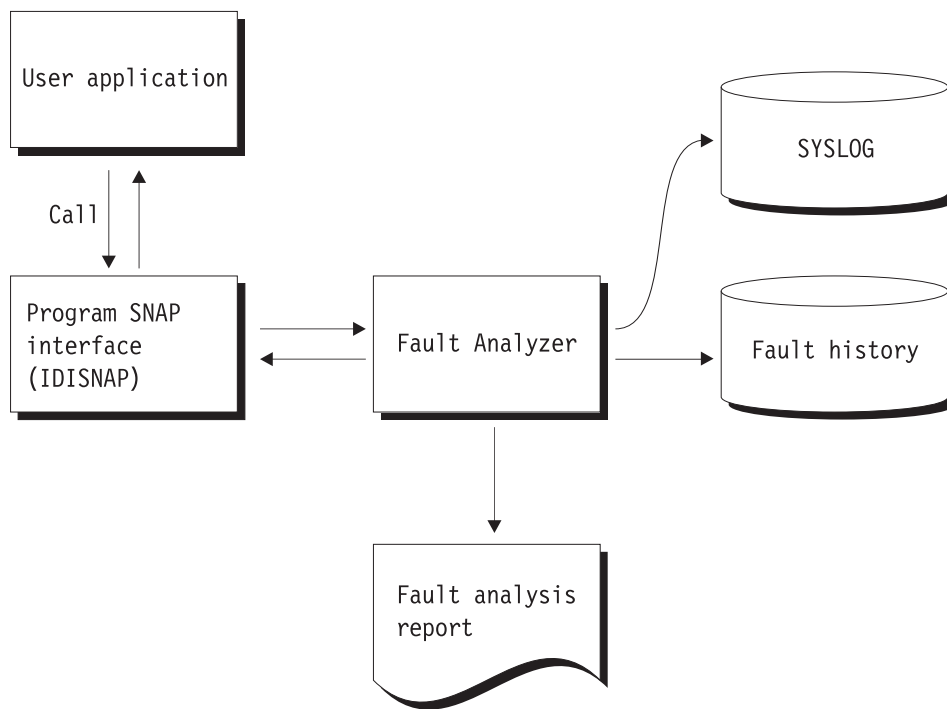


Figure 3. Real-time SNAP analysis

## 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 illustrates the reanalysis process.

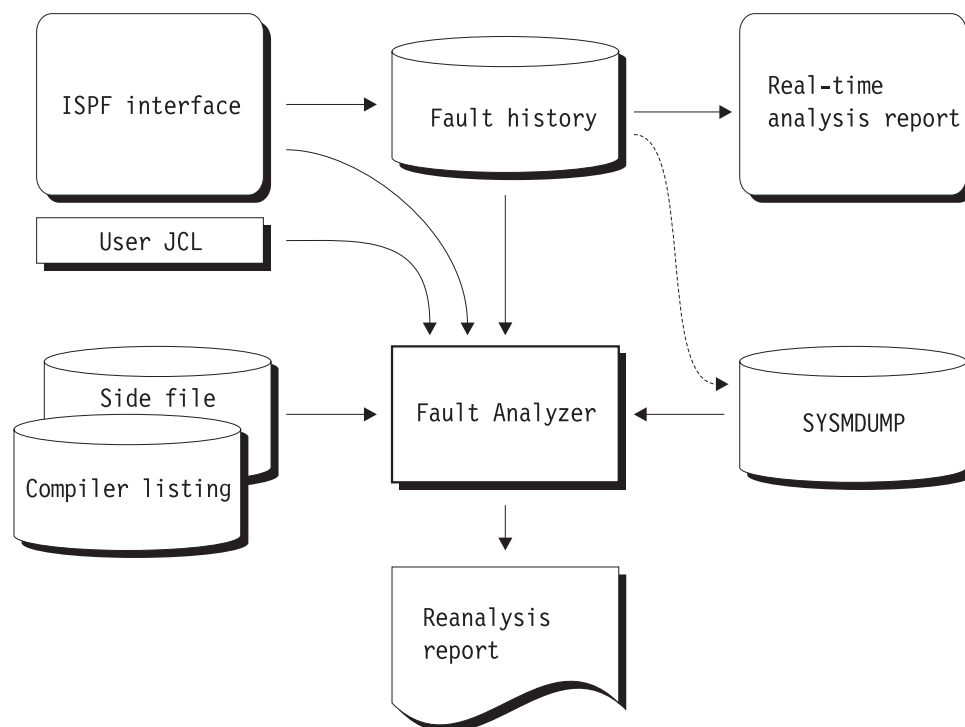


Figure 4. Reanalysis

If a SYSDUMP 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 SYSDUMP 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 31 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 a 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 can be changed using the IDIUTIL batch utility (see Chapter 27, "Managing history files (IDIUTIL utility)," on page 353 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>2</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. This allows Fault Analyzer to provide 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. This is described in Chapter 3, "The Fault Analyzer ISPF interface," on page 31.

A view member may be created in a data set identified by the IDIVIEWS DDname that contain the names of any number of history files that you wish to display simultaneously using the Fault Analyzer ISPF interface. For details, see "Using views" on page 35.

---

2. This 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 78.

## 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 will be rebuilt automatically the next time the history file is updated. This might, for example, be 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 will 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 or SetMaxFaultEntries control statements (for details, see Chapter 27, "Managing history files (IDIUTIL utility)," on page 353). If, for any reason, the \$\$INDEX member information is lost, then these settings will be automatically 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<sup>3</sup>
- 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
- WebSphere
- Java

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 does not yet perform analysis on applications using 64-bit addressing mode. C++ support is basic and does not provide any class information.

---

3. 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.

## Supported application environments

Only execution in home-space mode is supported.

Table 1. Application environment product versions supported by Fault Analyzer

Application environment		Fault Analyzer		
Product name	Version	V9.1	V10.1	V11.1
CICS/TS	V3R2 and earlier	GA	GA	GA
	V4R1	UK48699	GA	GA
	V4R2	-	-	UK68814
Enterprise COBOL	V4R1 and earlier	GA	GA	GA
	V4R2	UK52917	UK52832	GA
Enterprise PL/I	V3R7 and earlier	GA	GA	GA
	V3R8	UK47365	GA	GA
	V3R9	UK59799	UK59774	GA
	V4R1	UK65326	UK65177	UK65276
DB2	V9 and earlier	GA	GA	GA
	V10	UK62246	UK62342	GA
IMS	V11 and earlier	GA	GA	GA
	V12	UK62246	UK62342	GA

Table 1 legend:

Symbol	Meaning
GA	General availability—this is the base version of Fault Analyzer without any maintenance installed.
UKnnnnn	Earliest Fault Analyzer PTF level providing support.
-	No support is available on this version of Fault Analyzer at any maintenance level.

## Binder-related dependencies

In order to map CSECTs in load modules, Fault Analyzer utilizes the Application Programming Interface of the IBM Binder program. This generally occurs 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 245 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 makes it possible for Fault Analyzer to extract more information when you perform reanalysis.)

To provide a side file, you may 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 additional information, see Chapter 20, “Providing compiler listings or Fault Analyzer side files,” on page 303.

### 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 additional region size increases as the size and complexity of the abending program increases.

For detailed information about storage requirements, see “Storage recommendations” on page 218.

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 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 313 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 using the LE USRHDLR(CEEWUCHA) option.

If the LE USRHDLR(CEEWUCHA) option is in effect for an application abend that is analyzed by Fault Analyzer, and LE has passed control to this condition handler, then Fault Analyzer will suppress an initial U4038 LE abend to provide an Event Summary with the CEEWUCHA abend code as the important abend code (since the U4083 LE abend is suppressed).



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

If a condition handler with any name other than CEEWUCHA is specified in the LE USRHDLR option, then no suppression of the initial LE user abend is performed.

---

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

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

---

## 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 219), 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 16).
- A fault entry in a history file, which provides the ability to perform reanalysis of the fault.

A copy of the report written to JES is also included in the fault entry, and may 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 is the first step in the fault analysis process. In most cases, the analysis will be deemed satisfactory, and you will 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 will be written to the history file as a minidump, unless the MaxMinidumpPages option in effect specifies a lower limit.

**LOADER restriction:** Fault Analyzer will 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 319).

Suppression of dumps upon *successful completion*<sup>4</sup> of analysis is the default. However, when using

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

---

4. 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 has already been 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 additional 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 495).
- The RetainDump(ALL) option for non-CICS transaction faults (see “RetainDump” on page 495).
- An End Processing user exit (see “End Processing user exit” on page 402).

Dump suppression should not be confused with suppression of analysis or suppression of fault entry creation (see “Real-time exclusion processing” on page 25).

## 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 is that any subsequent dump-related CICS GLUE, for example XDUREQ, will then not be 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 495 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 specified in the parmlib config member, IDICNF00. This 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 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 (see “User options file IDIOPTS” on page 454 for more information).

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

### **RetainDump(ALL)**

Specify this option if you want to 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 will be 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 495 for more details.

The dump disposition part of this option is applicable to the use of the MVS IEAVTABX change options/suppress dump exit 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 468 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 458 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, although a SYSMDUMP DD statement is needed for normal SYSMDUMP processing in real time when using the MVS IEAVTABX change options/suppress dump exit, unless the IDITABD USERMOD is applied (see “Eliminating the need for a dump DD statement (++)IDITABD)” on page 245 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:

<b>IDILC</b>	PDS(E) data set containing C compiler listings
<b>IDILCOB</b>	PDS(E) data set containing COBOL compiler listings (other than OS/VS COBOL)
<b>IDILCOBO</b>	PDS(E) data set containing OS/VS COBOL compiler listings
<b>IDISYSDB</b>	PDS(E) data set containing COBOL or Enterprise PL/I SYSDEBUG, or XL C/C++ MDBG side files.
<b>IDILPLI</b>	PDS(E) data set containing PL/I compiler listings (other than Enterprise PL/I)
<b>IDILPLIE</b>	PDS(E) data set containing Enterprise PL/I compiler listings
<b>IDIADATA</b>	PDS(E) data set containing SYSADATA from Assembler compilations
<b>IDILANGX</b>	PDS(E) data set containing IDILANGX 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 303 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 17). The report is written to the IDIREPRT DDname, which is dynamically allocated to SYSOUT=*class*, if no prior allocation exists, and thus will be 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 will be used for the Fault Analyzer real-time report.

If you wish 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” on page 17.

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 183 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 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, the reports are preferred written to a single spool file, then this can be accomplished by adding an IDIREPRT DD statement to the job or startup procedure, as shown in the following example:

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

## Controlling the SYSOUT class of real-time reports

If no IDIREPRT allocation already exists, then Fault Analyzer will dynamically allocate IDIREPRT to SYSOUT=\*, or to the same SYSOUT class as the SYSUDUMP DDname. This can be changed 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
```

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 */
```

## The real-time analysis report

```
/* exit to suppress the analysis report.          */
/*****/
"IDIALLOC DD(IDIREPT) 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.

Suppressing real-time reports can be useful in, for example, a CICS environment, where the CICS job step might otherwise end up with hundreds of reports in the IDIREPT output.

---

## 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 219. This 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.

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:</p> <p><i>t nnn</i></p> <p>where:</p> <table><tr><td><i>t</i></td><td>Specifies the format of parameter 3 as either:</td></tr><tr><td>0</td><td>To indicate that a fixed-length 140 byte parameter area is used.</td></tr><tr><td>N</td><td>To indicate that a variable-length null-terminated parameter area is used.</td></tr></table> <p><i>nnn</i></p> <p>Specifies the number of parameters that follow parameter 1 as either:</p> <table><tr><td>000</td><td>To indicate that only parameter 1 is specified.</td></tr><tr><td>001</td><td>To indicate that parameters 1 and 2 are specified.</td></tr><tr><td>002</td><td>To indicate that parameters 1, 2, and 3 are specified.</td></tr><tr><td>00V</td><td>To indicate that parameters 1, 2, and 3 are specified, as well as one or more storage range address pairs in subsequent parameters.</td></tr></table> <p><b>Note:</b> Specifying 0000 in this parameter is the equivalent to invoking IDISNAP with register 1 set to zero.</p>	<i>t</i>	Specifies the format of parameter 3 as either:	0	To indicate that a fixed-length 140 byte parameter area is used.	N	To indicate that a variable-length null-terminated parameter area is used.	000	To indicate that only parameter 1 is specified.	001	To indicate that parameters 1 and 2 are specified.	002	To indicate that parameters 1, 2, and 3 are specified.	00V	To indicate that parameters 1, 2, and 3 are specified, as well as one or more storage range address pairs in subsequent parameters.
<i>t</i>	Specifies the format of parameter 3 as either:															
0	To indicate that a fixed-length 140 byte parameter area is used.															
N	To indicate that a variable-length null-terminated parameter area is used.															
000	To indicate that only parameter 1 is specified.															
001	To indicate that parameters 1 and 2 are specified.															
002	To indicate that parameters 1, 2, and 3 are specified.															
00V	To indicate that parameters 1, 2, and 3 are specified, as well as one or more storage range address pairs in subsequent parameters.															
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 will be 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>														

## Using the program SNAP interface (IDISNAP)

Table 2. IDISNAP input parameters (continued)

Parameter	Number of bytes	Description
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 498 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>
<p>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.</p> <p>Using the address range parameters is an alternative to, and will override, specification of the StorageRange option in parameter 3.</p> <p>To use these additional parameters, parameter 1 must be either 000V or N00V:</p> <ul style="list-style-type: none"><li>• If parameter 1 is 000V, then the area pointed to by parameter 2 must be 140 bytes, with any unused portion set to blanks.</li><li>• If parameter 1 is N00V, then the area pointed to by parameter 2 must be from 1 to 1024 bytes, with a null-character (X'00') immediately following the last character in the options string.</li></ul>		
Parameter <i>n</i>	4	Storage range begin address.
Parameter <i>n</i> +1	4	Storage range end address.

Options 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 four 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. As indicated by the DYNAM option in the CBL statement of the COBOL source, IDISNAP is in this example being called dynamically.

```

*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;
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);

```

## Using the program SNAP interface (IDISNAP)

```
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 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.

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

To facilitate calling the `com.ibm.faultanalyzer.Snap.Dump` method, the following is required:

- The Fault Analyzer provided `com.ibm.faultanalyzer.Snap` class must either exist in the application program execution path, or be available via the current `ClassPath`.
- The Fault Analyzer provided native API code in the `IDISWRAP` DLL must either exist in the application program execution path, or be available via the current `LibPath`.

The Fault Analyzer `com.ibm.faultanalyzer.Snap` class loads and calls the native API code in `IDISWRAP`.

If the above required files do not exist, then they can be installed as follows:

- Installing the `com.ibm.faultanalyzer.Snap` class
  1. Perform binary FTP transfer of `IDI.SIDIDOC1(IDIXJAVA)` to your HFS 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. If the HFS directory used is the same as the current Java program execution directory, then extract the files in the package using the command:
 

```
jar xvf IDIXJAVA
```
  - b. Add the directory and file name to the current `ClassPath`.

- Installing the `IDISWRAP` DLL
  1. Perform binary FTP transfer of `IDI.SIDIAUT2(IDISWRAP)` to your HFS directory as file `IDISWRAP`.

## Invoking Fault Analyzer from Java catch block

**Note:** The IDLSIDIAUT2(IDISWRAP) 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. Provide "execute" permission to the IDISWRAP DLL.
3. If the HFS directory used is not the current Java program execution directory, then add the directory to the current LibPath.

---

## 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 which 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 will mean that dumps will only be generated for four NullPointerException or SocketException throw events, and any more will be ignored. This is especially useful in cases where exceptions are caught and rethrown, but it does mean future Exceptions will not be 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, has been installed (see “Installing the MVS post-dump exit IDIXTSEL” on page 329), then a "skeleton" fault entry is created whenever an SVC dump is written. This 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 which caused the SVC dump to be written. If available, theabend code and abending program name is also provided. However, the first reanalysis of the dump registration fault entry will refresh the fault entry and save a report and minidump with it—for details, see “Refresh processing” on page 161.

---

## Real-time exclusion processing

While real-time analysis will never be 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 26.

## Real-time exclusion processing

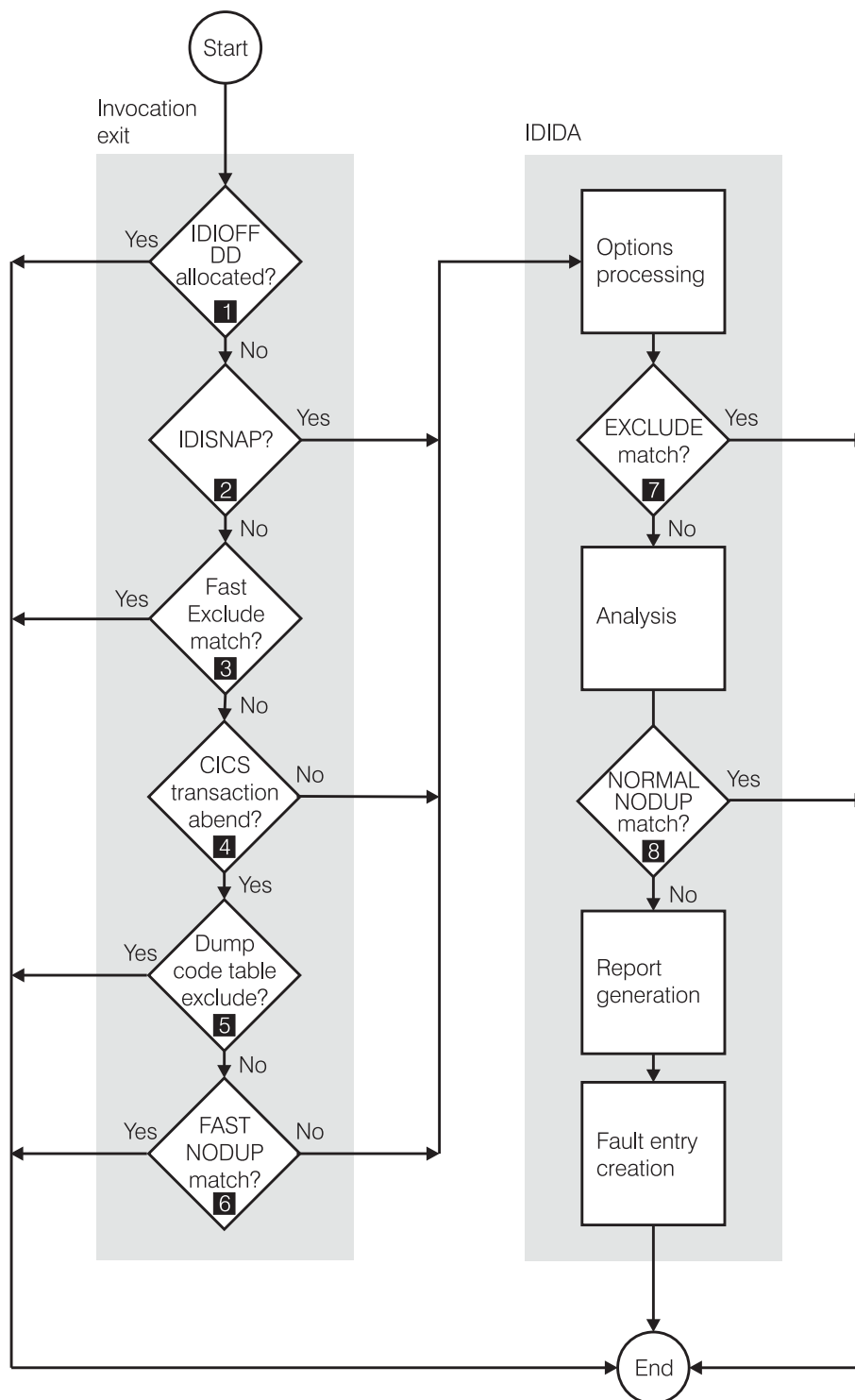


Figure 5. Real-time exclusion processing overview

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. This can be accomplished by, 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 additional information, see "Turning off Fault Analyzer using an environment variable (`_IDI_OFF`)" on page 368.

- 2** If calling IDISNAP from within your application (see "Using the program SNAP interface (IDISNAP)" on page 18), 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 270), then a matching Exclude option (see "Exclude/Include" on page 468) 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 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 468) 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.

## Real-time exclusion processing

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

## 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 sub-types, one at a CICS region level and one that encompasses an entire MVS image. The latter is applicable to IMS only.

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 32).

## 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 might, for example, be in the following situations:

- Insufficient storage. (Message IDI0005S issued—for details, see page 535.)
- Fault Analyzer abended. (Message IDI0047S issued—for details, see page 540.)
- Fault Analyzer timed out. (Message IDI0092S issued—for details, see page 546.)
- Invalid negative storage length request. (Message IDI0105S issued—for details, see 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 has been granted to use SDUMP as the recovery fault recording dump type, since this is the preferred dump type for performance reasons.

**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 222.

If SDUMP can not be used, then IEATDUMP will instead be 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 additional data set, into which MVS writes a dump of the address space. This 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 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 405.

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 (++IDISRFR)” on page 246. 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 228 for additional 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 will result in message IDI0128I to be issued. Refer to this message 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.IDIVPC0.

#### IEATDUMP dump title

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

**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(0x0000).

---

## 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 has been set to capture S0C4 abends on your system, then it is likely that unwanted matches will 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 additional SLIP trap as follows:

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

where *location* is LPAMOD if the IDIDA load module has been 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 89).
- Start an interactive reanalysis (for details, see Chapter 5, “Performing interactive reanalysis,” on page 97).
- 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 163).
- Analyze WebSphere or Java dumps (for details, see Chapter 8, “Performing Java analysis,” on page 175).

**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 194.

**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, will automatically be 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.

## The Fault Analyzer ISPF interface

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 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 303.

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 16 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 239.

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 will be 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, will update 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 33 shows an example of a Fault Entry List display:

```

File Options View Services Help
IBM Fault Analyzer - Fault Entry List
Command ==> Line 1 Col 1 80
Scroll ==> CSR

Fault History File or View : 'IBMUSER.DEMO.HIST'

{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).}

Fault ID Job/Tran User ID Sys/Job Abend Date Time
— F00323 IDIVPCOB IBMUSER MVS2 S0C7 2001/12/21 13:02:25
— F00445 ALLANT01 JACKIED MVS8 S0C7 2001/12/19 03:29:57
— F00444 ALLANT01 JACKIED MVS8 S0C7 2001/11/28 20:25:30
— F00442 ALLANT01 ALLANT MVS8 S0C7 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 S0C4 2001/08/02 17:03:18
— F00035 CICS53 n/a MVS2 n/a 2001/04/05 14:49:11
F1=Help F3=Exit F4=MatchCSR F5=RptFind F6=Actions F7=Up
F8=Down F10=Left F11=Right F12=MatchALL

```

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 means that you can place the cursor on such fields and press the Enter key to display additional information. For example, by selecting an abend code in the Abend or I\_Aband columns, the explanation for that abend code is displayed.

The history file or view (see “Using views” on page 35) 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 35.

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 34.



## The Fault Entry List display

```

Error
Line 1 Col 1 76
Command ==> Scroll ==> CSR
The following problems were found while processing the view in
DA.VIEWS(SWBAD1):

* -HistCols syntax error: Missing starting parenthesis. The -HistCols
  specification has been ignored.

* Data set 'xyz' open error: EDC5049I The specified file name could not be
  located.

* -Match syntax error: The subcommand entered for the "FRED" command was
  invalid. The -Match specification has been ignored.

Press PF3 to continue.

*** Bottom of data.

F1=Help    F3=Exit    F7=Up      F8=Down    F12=Cancel
```

Figure 7. Sample Error display

To exit from the error display, press PF3.

The error display will be 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 41. The default information displayed if no HistCols option has been specified and no customization has been made by the user is as shown in Figure 6 on page 33.

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 45.

As shown at the top of the display if help text is enabled (as illustrated in Figure 6 on page 33), 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 49. For information about how to show or hide help text, see “Adding or removing help text” on page 72.

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 will disappear, but will reappear again if the display is scrolled to the top. For general information about help text, see “Adding or removing help text” on page 72.

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

The line command input fields on the left side of the display will remain 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, or one 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 58). 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.

Apart from being able to see fault entries from multiple history files simultaneously, views can also be used to provide a specific column layout for the Fault Entry List display (see “Specifying a default column layout” on page 252), or to provide a selection criteria for the initially displayed list of fault entries (see “Specifying an initial fault entry selection criteria” on page 252).

For information about how to set up views, see “Setting up views” on page 250.

## 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, you may type its name on the “Fault History File or View” line as the example shown at **1** in Figure 8 on page 36 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 Fault Entry List display

```

File Options View Services Help
-----
IBM Fault Analyzer - Fault Entry List                               Line 1 Col 1 80
Command ==> _____ Scroll ==> CSR

Fault History File or View : 'my.hist' 1 _____

{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).}

  Fault ID Job/Tran User ID Sys/Job Abend Date      Time
  ---
  F00323 IDIVPCOB IBMUSER MVS2   S0C7 2001/12/21 13:02:25
  F00445 ALLANT01 JACKIED MVS8   S0C7 2001/12/19 03:29:57
  F00444 ALLANT01 JACKIED MVS8   S0C7 2001/11/28 20:25:30
  F00442 ALLANT01 ALLANT  MVS8   S0C7 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 S0C4 2001/08/02 17:03:18
  F00035 CICS53   n/a      MVS2   n/a   2001/04/05 14:49:11
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 wish 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 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 may 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
<b>'FRED.*'</b>	All history file names with FRED as the first qualifier and at least one more qualifier.
<b>'FRED.**'</b>	All history file names with FRED as the first qualifier.
<b>'FRED.**.HIST'</b>	All history file names with FRED as the first qualifier and zero or more qualifiers following.
<b>'AAA%*.B*%%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 58). This will bring up the Last Accessed Fault History Files or Views display as the example shown in Figure 9 on page 38.

## The Fault Entry List display

```

File Options View Services Help
s      Last Accessed Fault History Files or Views
I
C      Enter the number corresponding to one of the following
F      previously accessed history files or views and
      press Enter:
      1. 'IBMUSER.DEMO.HIST'
      2. 'IBMUSER.HIST'
      3. 'DA.DCAT'
      4. (APC) Sample view of APC history files
      5.
      6.
      7.
      8.
      9.
      10.

      F1=Help   F3=Exit   F12=Cancel

F00050 PSTRANDR PSTRAND STPLEX4B S0C4 2001/08/02 17:03:18
F00035 CICS53  n/a     MVS2    n/a   2001/04/05 14:49:11
F00034 CICS53  n/a     MVS2    S08E 2001/03/22 13:12:23
F1=Help   F3=Exit   F4=MatchCSR F5=RptFind F6=Actions F7=Up
F8=Down   F10=Left  F11=Right F12=MatchALL

```

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.

### 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 58). This will bring up the Last Accessed Fault History File Entries display as the example shown in Figure 10 on page 39.

```

File Options View Services Help
s      Last Accessed Fault History File Entries
I
C      Enter the number corresponding to one of the following
      previously accessed history file entries and press Enter:
F      1. 'IBMUSER.DEMO.HIST(F00323)'
      2. 'IBMUSER.DEMO.HIST(F00345)'
{      3. 'IBMUSER.DEMO.HIST(F00348)'
r      4. 'IBMUSER.HIST(F00331)'
      5.
      6.
      7.
      8.
      9.
      10.
      F1=Help    F3=Save    F4=Reset    F7=Up    F8=Down    F10=Left
      F11=Right  F12=Cancel

      F00345 CS01    CICSUSER CSCB0010 AEIL  2001/08/23 07:43:35
      F00050 PSTRANDR PSTRAND  STPLEX4B S0C4  2001/08/02 17:03:18
      F00035 CICS53   n/a      MVS2     n/a    2001/04/05 14:49:11
      F00034 CICS53   n/a      MVS2     S08E   2001/03/22 13:12:23
      F1=Help    F3=Exit    F4=MatchCSR F5=RptFind F6=Actions F7=Up
      F8=Down    F10=Left   F11=Right  F12=MatchALL

```

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 58). This will bring up a display as shown in Figure 11 on page 40.

## The Fault Entry List display

```

File Options View Services Help
S      View List
I      View List
C      Command ==> Line 1 Col 1 76 80
      Scroll ==> CSR
F      Line commands: S (select) B (browse).
{
r      - Name      Description
      - APC        Sample view of APC history files
      - BATCH      Batch History Files
      - DB2        SAMPLE VIEW OF DB2 HISTORY FILES
      - JOHNS      View of CICS and IMS History files
      - SUBS       SubSystem History Files
      - SW         Test
      - TOM        Tom's Views
      - VIEW2      TEST VIEW
      - ZZZZZZZZ   (No description available)

      *** Bottom of data.

      F1=Help    F3=Exit    F7=Up    F8=Down    F12=Cancel
F8=Down    F10=Left    F11=Right

```

Figure 11. Sample View List display

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

- B** This will permit 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.

```

File Options View Services Help
S      View List
I      File Browse
C      'DA.VIEWS(JOHNS)' Line 1 Col 1 76
      Command ==> Scroll ==> CSR
F      * View of CICS and IMS History files
{      CTEST.DUMPA.DACICS.DCAT
r      CTEST.DUMPA.DAIMS.DCAT

      B      *** Bottom of data.

      *

      F1=Help    F3=Exit    F7=Up    F8=Down    F12=Cancel
F8=D

```

Figure 12. Sample File Browse display

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

- S** This will select 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 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 33.

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 58). This will bring up the Fault Entry List Column Configuration display as the example shown in Figure 13.

File View Services Help

Fault Entry List Column Configuration
Line 1 Col 1 80

Command ==>
Scroll ==> CSR

Current Fault Entry List Column Configuration (Sample Data):

Fault ID	Job/Tran	User ID	Sys/Job	Abend	Date	Time
F00249	IDIVPCOB	FRED	MVSA	S0C7	2001/11/22	15:29:03

Column Configuration Settings:

{Below, you may change your Fault Entry List display column configuration. 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. The resulting column configuration will be shown above.}

Order	Column
1	Fault ID
2	Job/Tran
3	User ID
4	Sys/Job

Default column configuration used

1

F1=Help
F3=Sa
F8=Down

F10=Left
F11=Right
F12=Cancel

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 permits you to see which of the selected columns that will be 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.

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.

## The Fault Entry List display

After pressing Enter, the resulting column configuration will be 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 will default 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 41). There are four different possibilities:

- **Default column configuration used**

This 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 will be 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 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 will replace 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 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 will be 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 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 will replace 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 will 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:



<b>Fault_ID</b>	The ID assigned to the fault.
<b>Abend</b>	<p>The initial (if more than one) abend code.</p> <p>For a fault entry created using IDISNAP, the abend code is shown as "SNAP".</p>
<b>Appl_ID</b>	The CICS application ID.
<b>CICS_Trn</b>	The failing CICS transaction ID. This column is only applicable to CICS.
<b>Class</b>	The job class in which the job was executing.
<b>Date</b>	The date when the fault occurred in LOCALE-option dependent format.
<b>Dups</b>	The number of duplicate faults detected. See “NoDup” on page 482 for details about duplicate determination.
<b>Dup_Count</b>	Deprecated—use Dups instead.
<b>Dup_Date</b>	The date when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this is set to the initial abend date (see Date).
<b>Dup_Time</b>	The time when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this is set to the initial abend time (see Time).
<b>EXEC_Pgm</b>	The abending job step program name from the JCL EXEC PGM= parameter.
<b>Group_ID</b>	The security server default group ID.
<b>History_File_DSN</b>	The history file data set name containing the displayed fault entry.
<b>I_Abend</b>	<p>The abend code for which Fault Analyzer was invoked.</p> <p>For a fault entry created using IDISNAP, the abend code is shown as "SNAP".</p>
<b>IMS_Pgm</b>	The IMS application program name for faults involving IMS.
<b>Job/Tran</b>	For a CICS transaction abend, this 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.
<b>Job_ID</b>	The JES job ID of the abending job.
<b>Job_Type</b>	<p>The abending job type as one of the following:</p> <p><b>Batch</b> Batch job</p> <p><b>CICS</b> CICS transaction</p> <p><b>DumpReg</b> Dump registration</p> <p><b>STC</b> Started task</p> <p><b>TSO</b> TSO user</p>
<b>Jobname</b>	The name of the abending job.
<b>Lock</b>	The fault entry lock flag. For information about the purpose of this flag, see “Viewing fault entry information” on page 78.

## The Fault Entry List display

<b>Minidump</b>	Indication of minidump availability as follows: <b>Yes</b> Minidump is available <b>No</b> Minidump is not available
<b>Module</b>	The point-of-failure module name.  For dump registration fault entries, this 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 is updated to become the point-of-failure module name, which might be different.
<b>MD_Pages</b>	The number of minidump pages saved for the fault.
<b>MVS_Dump</b>	Indication of MVS dump availability as follows: <b>Yes</b> MVS dump is available <b>No</b> MVS dump is not available  <b>Note:</b> This column does not take into consideration if the associated MVS dump data set has been deleted or does not exist on the system on which the fault entry is being displayed. It only indicates if an MVS dump data set was available at the time of abend.
<b>MVS_Dump_DSN</b>	The name of any associated MVS dump data set written at the same time as when the fault occurred.  <b>Note:</b> This column does not take into consideration if the associated MVS dump data set has been deleted or does not exist on the system on which the fault entry is being displayed.
<b>Netname</b>	CICS transaction netname.
<b>Offset</b>	The point-of-failure offset.
<b>Program</b>	The point-of-failure program name.
<b>Stepname</b>	The job step name of the abending job.
<b>Sys/Job</b>	For a CICS transaction abend, this 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.
<b>System</b>	The system ID on which the abend occurred.
<b>Task</b>	The failing CICS transaction task number. This column is only applicable to CICS faults.
<b>Term_ID</b>	CICS transaction terminal ID.
<b>Time</b>	The time when the fault occurred in LOCALE-option dependent format.
<b>Tran_ID</b>	Deprecated—use CICS_Trn instead.
<b>User_ID</b>	The user ID associated with the abending job/transaction.
<b>User_Title</b>	User-maintained title information (see “Viewing fault entry information” on page 78 for details).

**Username** User-maintained name information (see “Viewing fault entry information” on page 78 for details).

## Sorting and matching fault entries

The Fault Entry List display column headings are tabable 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 which satisfy a given MATCH criteria.

A sample Column Attributes display is shown in the following.

File Options View Services Help

Column Attributes

Column Name:  
Dup\_Date

Sort:  
Enter "/" to select  

Ascending  
Descending

Match:  

\*\_\_\_\_\_

try List
Line 28 Col 1 80  
Scroll ==> CSR

(SW) SWILKEN.HIST\*

Dup_Time	Jobname	Abend	Module	System	User_I
15:57:37	PLICBPL	SNAP	PLICBPL	FAE1	SWILKE
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
13:44:02	CSCB0650	SNAP	CSCB0650	FAE1	TRUONG
12:50:07	IDIVPCOB	S0C7	IDISCBL1	FAE1	SWILKE
12:08:10	IDIVPCOB	U4039	IDISCBL1	FAE1	SWILKE
12:07:19	IDIVPCOB	S0C7	n/a	FAE1	SWILKE
12:06:36	IDIVPCOB	S0C7	IDISCBL1	FAE1	SWILKE
09:30:49	PLI23	U4000	@960IDI	FAE1	SWILKE
08:53:50	PLI23	U4000	@960IDI	FAE1	SWILKE
11:07:26	PLI23	U4000	@960IDI	FAE1	SWILKE
11:06:22	PLI23	S806	n/a	FAE1	SWILKE
11:05:16	PLI23	U4000	@960IDI	FAE1	SWILKE
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.

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 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 which match the specified string will be 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 Fault Entry List display

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 45), 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 will update 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 on page 33 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:

```

File Options View Services Help
IBM Fault Analyzer - Fault Entry List
Command ==> _____ Scroll ==> CSR
1 3 of 319 rows matched

Fault History File or View : 'IBMUSER.DEMO.HIST'

{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).}

2
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

*** Bottom of data.

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 will be issued that shows how many rows, of the ones previously displayed, that matched the selected value. This message will only remain until a function key or the Enter key is pressed.
- 2 Column headings on which a MATCH is currently active will be highlighted. These will remain so 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:

## The Fault Entry List display

```
File Options View Services Help
IBM Fault Analyzer - Fault Entry List                2 of 3 rows matched
Command ==> _____ Scroll ==> CSR

Fault History File or View : 'IBMUSER.DEMO.HIST'

{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).}

  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

*** Bottom of data.

F1=Help      F3=Exit      F4=MatchCSR  F5=RptFind   F6=Actions   F7=Up
F8=Down      F10=Left     F11=Right   F12=MatchALL
```

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 will be performed to show only those fault entries that have an abend value of S0C1.

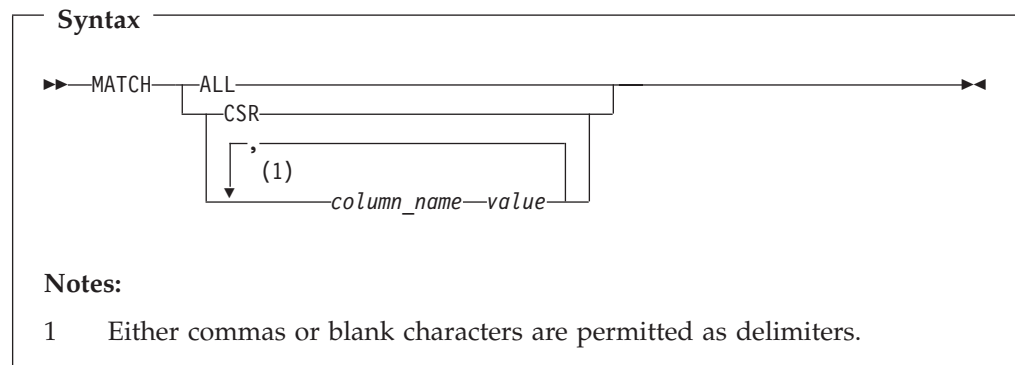
Wildcard characters can be used to specify generic MATCH values. The supported wildcard characters are an asterisk (\*) to indicate zero, one or more characters, and a percent sign (%) to indicate only a single character.

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.

### 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 42, for example, Abend or IMS\_Pgm. Neither the column name, nor the specified value, are 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 is not the same as doing a REFRESH, which looks at the history file, and so may 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 46).

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

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.

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. The available actions are:

## The Fault Entry List display

### **B Batch reanalysis**

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

See Chapter 4, “Performing batch reanalysis,” on page 89 for details.

### **C Copy**

Copy the fault entry to a different history file.

See “Copying history file entries” on page 86 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 76 for details.

### **H Duplicate history**

When available, shows details about faults which have occurred, 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 83 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 97 for details.

### **M Move**

Move the fault entry to a different history file.

See “Moving history file entries” on page 87 for details.

### **V (or S)**

#### **View report**

View the saved fault analysis report:

See “Viewing a saved report” on page 70 for details.

### **X XMIT**

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

See “Transmitting history file entries” on page 87 for details.

### **? View fault entry information**

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

See “Viewing fault entry information” on page 78 for details.

If you enter a line command against an entry, and Fault Analyzer is unable to complete the command (for example, you attempt to run a batch dump reanalysis



against a fault that has no associated dump data set, or the dump data set is unavailable), then the line command is not cleared from the line.

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. If Fault Analyzer is unable to honor a command, then it stops processing. It clears the line commands from each entry it was able to process, but leaves the line commands for each entry it failed to process or the entry at which it could not 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 58). This will bring up the Fault History File Properties display as the example shown in Figure 17.

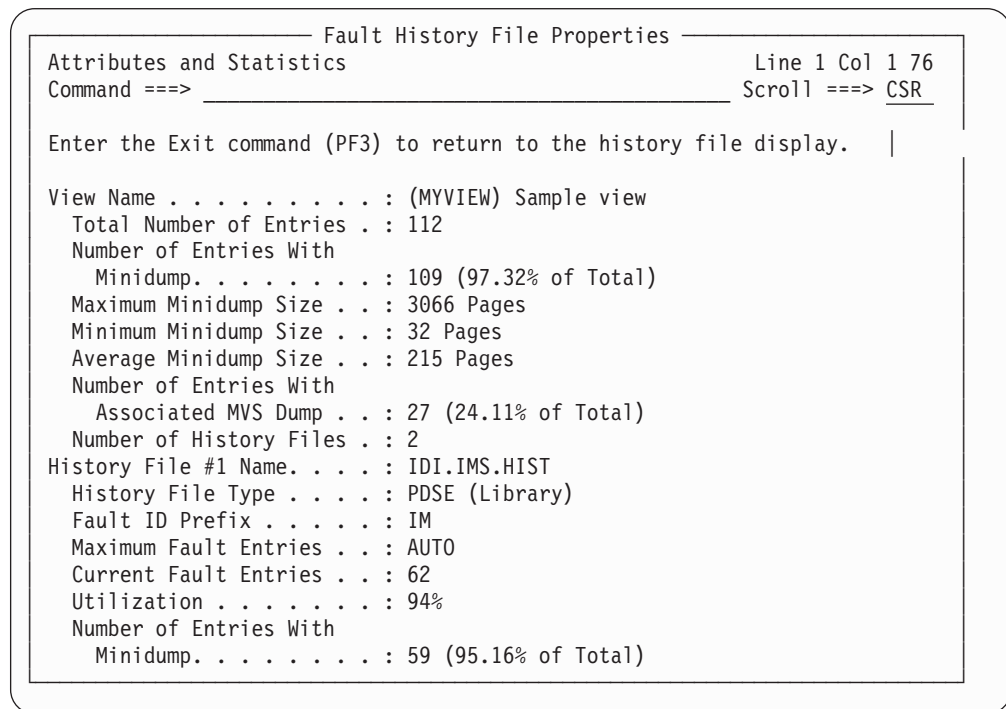


Figure 17. Sample Fault History File Properties display

The following information is available from this display:

### 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 353).

## The Fault Entry List display

### History File Access

Read or Update.

### History File Name

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

### History File Type

This is indicated as one of the following:

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

### Maximum Fault Entries

The maximum number of fault entries to be maintained in this history file, before automatic deletion of the oldest entries will occur. The maximum fault entries can be changed with the IDIUTIL batch utility SetMaxFaultEntries control statement (see Chapter 27, "Managing history files (IDIUTIL utility)," on page 353).

If "*nnn*", then this is the maximum number of fault entries which can be maintained in the PDS(E) history file. Additional data set extents will be allocated as needed to achieve this number of fault entries. An out-of-space condition will occur 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 "*nnn*,ThenAUTO", then the PDSE history file will be maintained automatically once a minimum of *nnn* 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 additional data set extents will generally be allocated and no out-of-space conditions are expected.

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

### 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 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 58). This will bring up the New History File Allocation display as the example shown in Figure 18.

New History File Allocation
Line 1 Col 1 76

Command ==> \_\_\_\_\_
Scroll ==> CSR

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

History File Name . . . . : \_\_\_\_\_  
Primary Space . . . . . : 100      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 will be 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 will not be available for input, but will show the Move/Copy target history file name.

### Primary Space

The number of cylinders to allocate as the primary space. The default is 100.

### Secondary Space

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

## The Fault Entry List display

### 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 will occur. This must be a number 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 58).

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

Next, the Change Fault History File Settings display, as the example shown in Figure 19 on page 55, is presented.

```

e Change Fault History File Settings                               Line 1 Col 1 76
Command ==> _____ Scroll ==> CSR

History File Name . . . . : MY.HIST
Data Set Type . . . . . : PDSE (Library)
Current Number of Fault
Entries . . . . . : 321

Press Enter to change history file settings, or press PF3/PF12 to cancel.

Fault Entry Prefix. . . . : F      (1-3 alphabetic characters)
Minimum Fault Entries (*) : 25     (25-9999999)

(*) When the total number of fault entries exceeds this value, then the
    history file is auto-managed.

*** Bottom of data.
  
```

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 appended fault entry number, forms the fault ID for this history file. All new fault entries created in the history file will be 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 will occur. This must be a number 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 Fault Entry List display

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 will always be 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 and SetMaxFaultEntries 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 226.

## 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 58).

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 will show 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 re-read the history file or view to include any such entries, you may 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 58).

The refresh performed in this manner will cause 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 will be 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 re-read 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 which 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 will be limited to the fault entries currently in the \$\$INDEX member of the history file data set, but will 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 will become 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** ):

File Options View Services Help

IBM Fault Analyzer - Fault Entry List

Refresh complete

Command ==>

Scroll ==> CSR

Fault History File or View : (ALLTEST) All testing

6 history files might have updates pending. 1

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:

## The Fault Entry List display

File Options View Services Help

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 will generally be 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 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 78).

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 512).

---

## 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 will be 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 will be 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** Is the name of the action-bar pull-down menu at the top of the display.

**menu\_option** Is the name of the available option on the first and any subsequent menus.

The available pull-down menu options are shown 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 page 163), or Java dump analysis (see page 175).

**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 54 for additional 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 56 for additional information.

**File->Exit**

Selecting this option is the equivalent to issuing the ISPF EXIT command. Generally, this 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 bring up the Fault Entry Information display as shown in Figure 29 on page 79. Selecting this option is identical to issuing the INFO command (see “INFO” on page 65) 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 49).

**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 51 for additional information.

## Action-bar pull-down menus

### **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 173 for additional information.

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

Used to display a list of up to ten previously accessed history file entries. See "Changing the history file or view displayed" on page 35 for additional information.

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

Used to display a list of up to ten previously accessed history files or views. See "Changing the history file or view displayed" on page 35 for additional information.

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

Used to display a list of all available views. See "Changing the history file or view displayed" on page 35 for additional information.

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

Permits the allocation of a new history file. See "New history file allocation" on page 53 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 76 for additional information.

### **Options->Batch Reanalysis Options**

Used to set options for batch reanalysis. See "Batch reanalysis options" on page 89 for additional 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 97 for additional 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 75 for additional 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 140 for additional information.

### **Services->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 73 for additional information.

### **View->Add Blank Lines**

Used to format displays using blank lines when appropriate to separate the information shown. This is the default. See "Adding or removing blank lines" on page 72 for additional information.

### **View->Add Help Text**

Used to add explanatory text to displays which might assist the

inexperienced user. This is the default. See “Adding or removing help text” on page 72 for additional information.

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

Used to add pseudo assembler instructions to the Compiler Listing display. See “Displaying source code” on page 135 for additional information.

### **View->Column Configuration**

Used to change the columns of information shown for faults listed on the Fault Entry List display. See “Fault entry list column configuration” on page 41 for additional information.

### **View->Preferred formatting Width**

Used to set the preferred display formatting width. See “Setting preferred formatting width” on page 72 for additional information.

### **View->Refresh**

Used to re-read all entries in the selected history file or view. See “Refreshing fault entry information” on page 56 for additional 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 72 for additional information.

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

Used to remove explanatory text from displays. See “Adding or removing help text” on page 72 for additional information.

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

Used to remove pseudo assembler instructions from the Compiler Listing display. This is the default. See “Displaying source code” on page 135 for additional information.

---

## Commands

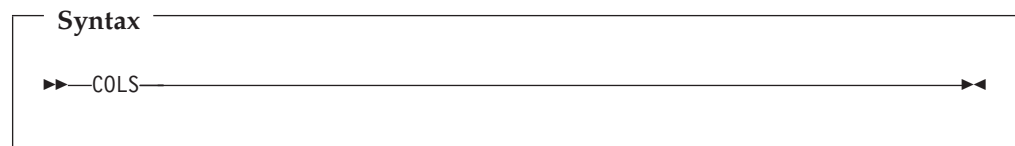
From the Fault Analyzer ISPF interface, the following primary commands are available:

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

## COLS

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

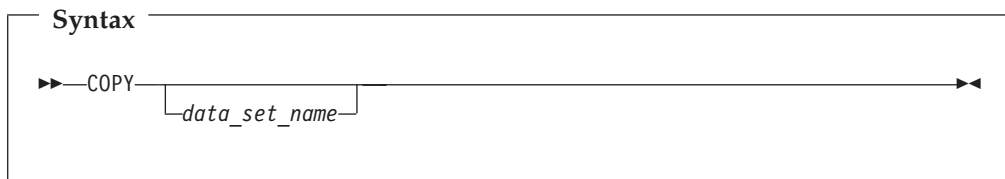
This command is only available from the Fault Entry List display.



For additional information, see “Fault entry list column configuration” on page 41.

## COPY

Copies the current display to the specified data set name.



where:

### **data\_set\_name**

Is 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.

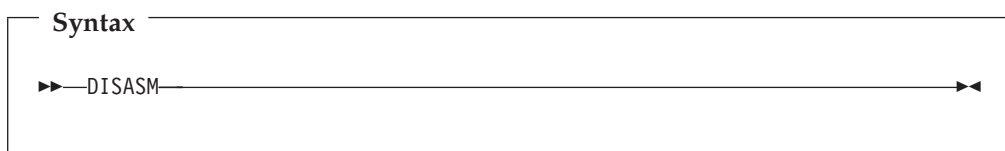
If a sequential data set name is specified, and the data set does not exist, then it will be created using the attributes: RECFM=VB LRECL=1028 BLKSIZE=6144. The attributes of any existing data sets will not be modified.

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

For additional information about the COPY command and an example of its use, see “Copying interactive displays to a file” on page 75.

## 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.



For information about how to use this command, see “Disassembling object code” on page 150.

## 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 145.

**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 83.

**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**

Is 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**

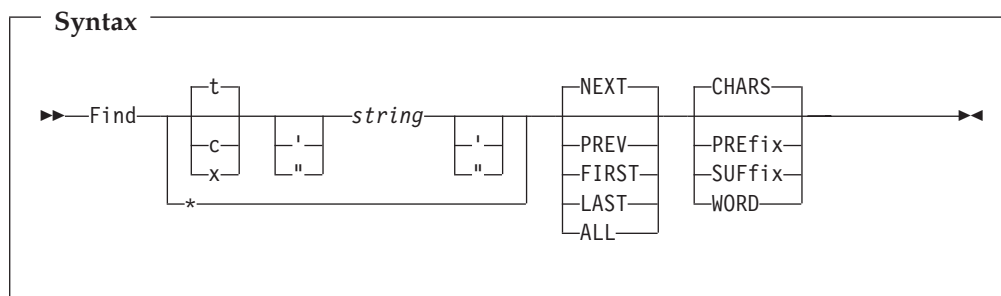
Are optional parameters to be passed to the Formatting user exit.

For information on creating your own Formatting user exits, and making these available to your environment, see “User-specific report formatting” on page 154.

**FIND**

Used to locate a text string in the current display.

## FIND



where:

<b>t</b>	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.
<b>c</b>	Indicates that a character string is being searched for. Character strings are case-sensitive.
<b>x</b>	Indicates that a hexadecimal value is being searched for. An even number of digits must be specified.
<b>string</b>	Is 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.
<b>*</b>	Indicates that the same string as previously searched for is to be used.
<b>NEXT</b>	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.
<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 will be considered a match. This 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 may 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 may be specified.

**WORD** 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 issued from the Dump Storage display<sup>5</sup> behaves different to the FIND command issued from all other displays. This 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 may 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 58), 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 49). In either case, the Fault Entry Information display, as shown in Figure 29 on page 79, is presented.

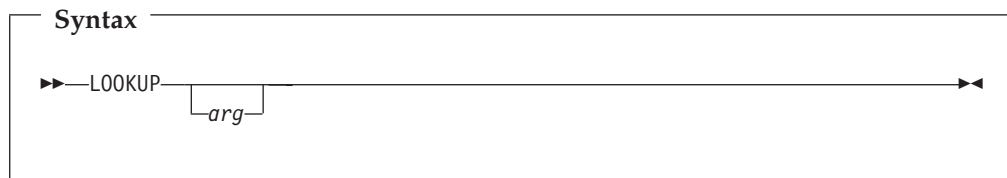
Syntax

▶▶—INFO—▶▶

5. 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.

## LOOKUP

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



where:

**arg** Is 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:

- |                 |  |
|-----------------|--|
| <b>XYZ</b>      | Implicitly searches for a match on *XYZ*, which means that zero or more characters may precede XYZ, and zero or more characters may follow (Eg. XYZ, AXYZ, XYZB, AXYZBBB). |
| <b>A*</b>       | Matches everything starting with A, followed by zero or more characters (Eg. A, AB, ABA000I).  |
| <b>A%</b>       | Matches everything starting with A, followed by exactly one character (Eg. AA, AB, AC).  |
| <b>ABA%%%%I</b> | Matches everything starting with ABA, following by four characters and finishes with I (Eg. 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 can be specified.

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 239.

For additional information about the LOOKUP command, and an example of its use, see “Displaying user-selected message or abend code explanations” on page 73.

## MATCH

Refer to “Additional ways to match and select faults” on page 46 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 will display 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.

<p><b>Syntax</b></p> <p>▶▶—NEXT—▶▶</p>
--

## 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.

<p><b>Syntax</b></p> <p>▶▶—NOTELIST—▶▶</p>
--

For additional information, see “Creating and managing user notes” on page 140.

## 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 will display 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.

<p><b>Syntax</b></p> <p>▶▶—PREV—▶▶</p>
--

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

### Syntax

►►—QUIT—◄◄

## REFRESH

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

Causes the current history file or view to be re-read 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 148.

**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
```

A 64-bit address can be specified by typing an underscore between bits 31 and 32. The value on the right hand side of the underscore is automatically padded with leading zeroes to form 8 digits. For example, to display the storage at address 1\_00100000, enter:

```
SHOW 1_100000
```

If the SHOW command is issued without specifying an address, then either 0, or the storage address 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
```

## SHOW

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

## 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 152.

## 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 39).

### Syntax

►►—VIEWS—◄◄

---

## Viewing a saved report

To view the saved report associated with a fault, enter **V** (or **S**) against the entry in the history file display. Figure 22 on page 71 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> EVENT 1 OF 5: CALL (DSA ADDRESS 00034018)
+ <H2> EVENT 2 OF 5: CALL (DSA ADDRESS 000340E0)
+ <H2> EVENT 3 OF 5: CALL (DSA ADDRESS 000341D0)
+ <H2> EVENT 4 OF 5: CALL (DSA ADDRESS 00034390)
+ <H3> Associated Storage Areas
_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 will automatically be 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 'I' line command had been used. This 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 pressed, the section will be 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 pressed, the section will be 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 will be “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 58).

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 58).

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 is the width that Fault Analyzer should use whenever possible during formatting of information for displays.

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 58).

File Options View Services Help

Preferred Formatting Width Specification

Specify the display formatting width to be used by Fault Analyzer whenever possible and press Enter.

Preferred Formatting Width 80 (Minimum 80)

F1=Help F3=Exit F12=Cancel

Fault ID	Job/Tran	User ID	Sys/Job	Abend	Date	Time
F00323	IDIVPCOB	IBMUSER	MVS2	S0C7	2001/12/21	13:02:25
F00445	ALLANT01	JACKIED	MVS8	S0C7	2001/12/19	03:29:57
F00444	ALLANT01	JACKIED	MVS8	S0C7	2001/11/28	20:25:30
F00442	ALLANT01	ALLANT	MVS8	S0C7	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	S0C4	2001/08/02	17:03:18
F00035	CICS53	n/a	MVS2	n/a	2001/04/05	14:49:11
F00034	CICS53	n/a	MVS2	S08E	2001/03/22	13:12:23

F1=Help F3=Exit F4=MatchCSR F5=RptFind F6=Actions F7=Up  
F8=Down F10=Left F11=Right F12=MatchALL

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, which may be issued from the interactive fault history file display or the interactive fault reanalysis report, 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 66.)

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 ==> <u>CSR</u>
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 ==> <u>CSR</u>
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 66), or you may 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 74 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
    - .
    - . (not all abend codes shown)
    - .
    - U3469
    - U3498
    - U3499
    - U3999
    - U4095
  - + Language Environment User Abend Codes
  - + CICS User Abend Codes
  - + MVS Abend Codes
- + Messages
- + Miscellaneous Information

Only information that is indexed by Fault Analyzer can be entered—see “LOOKUP” on page 66 for a list of these.

---

## 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 62.)

For example, if entering the command:

```
COPY PRINT(A)
```

the current display will be 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 58). This brings up the About Fault Analyzer display as the example shown in Figure 26.

```
----- About Fault Analyzer -----
Copyright and General Usage Information                               Line 1 Col 1 76
Command ==> _____ Scroll ==> CSR

IBM Fault Analyzer for z/OS V12R1M0 (MVS 2012/01/01)

Licensed materials - Property of IBM(*) 5655-W69

(C) Copyright IBM Corp. 2000, 2012. All rights reserved.

US government users restricted rights - use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.

Materials displayed or reproduced by this program may be protected by
copyright or contract restrictions of IBM and/or others. The user is
responsible for having permission to display or reproduce such materials
and for including applicable copyright notices and legends.

If any IBM machine-readable documentation is accessed or reproduced by or
through this program, IBM grants limited permission to licensees of the
IBM machine-readable documentation to make hardcopy or other reproductions
F1=Help   F3=Exit   F5=RptFind F7=Up       F8=Down   F10=Left
F11=Right
```

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 additional information about the lock flag, and how to change its value, see “Viewing fault entry information” on page 78.

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 58). This will bring up the Fault Analyzer Preferences display as illustrated in Figure 27.

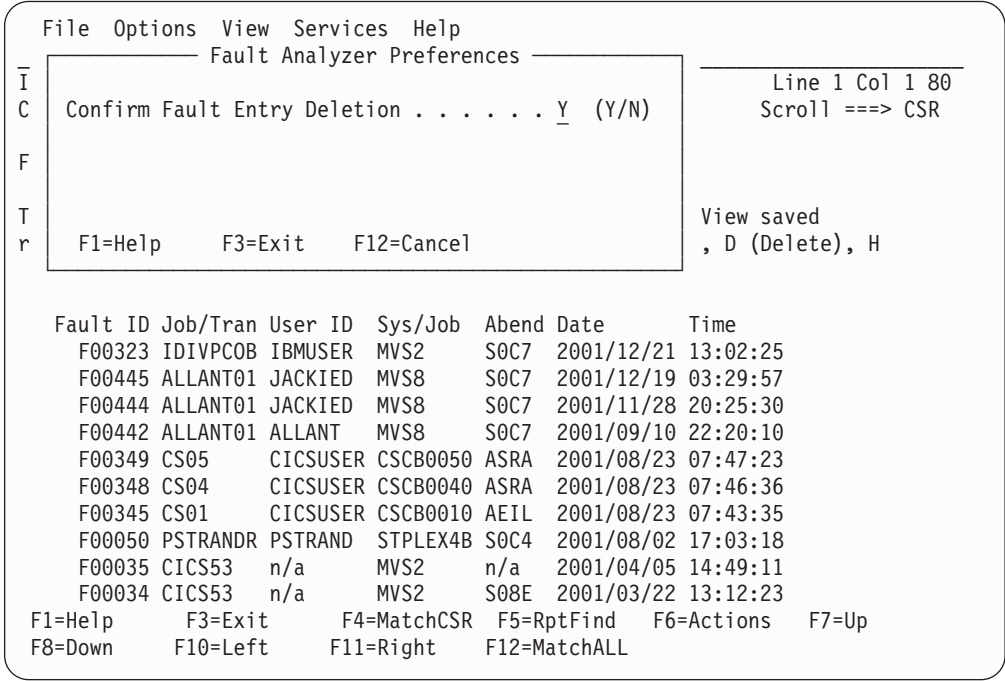


Figure 27. Sample Fault Analyzer Preferences display

The option you can set is:

### Confirm Fault Entry Deletion

If this 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

Command ==> \_\_\_\_\_ Scroll ==> CSR

Press Enter to delete the selected fault entries, or press PF3/PF12 to abort the delete request.

Keep delete confirmation on : Y (Y/N)

Selected fault entries:  
SW00882 SW00886 SW00926 SW00927 SW16070 SW16072

\*\*\* Bottom of data.

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

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 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 additional 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 65) or by selecting “Fault Entry Information” from the action-bar “File” pull-down menu (see “Action-bar pull-down menus” on page 58).

An example of the fault entry information display follows:

File View Services Help	
Fault Entry Information	
Command ==>	Line 1 Col 1 80 Scroll ==> CSR
Fault ID. . . . . : BAT00009 User Name . . . . . : _____ User Title. . . . . : _____ Lock Flag . . . . . : (Not locked) Abend Code. . . . . : S0C4 POF Module Name . . . . . : IBMBLIIA POF Program Name. . . . . : IBMBLI11 POF Offset. . . . . : 96C Abend Date. . . . . : 2003/09/22 Abend Time. . . . . : 15:03:02 Job Name. . . . . : PLI23 Job ID. . . . . : JOB30836 Job Execution Class . . . . : A Job Type. . . . . : Batch Job Step Name . . . . . : G0 EXEC Program Name . . . . : @960IDI User ID . . . . . : RTURNER System Name . . . . . : FAE1 Application ID. . . . . : n/a CICS Transaction ID . . . . : n/a CICS Task Number. . . . . : n/a CICS Terminal ID. . . . . : n/a CICS Terminal Netname . . . : n/a IMS Program Name. . . . . : n/a Accounting Information. . . : n/a Duplicate Count . . . . . : 2 Minidump Pages. . . . . : 60 History File Data Set Name. : DA.DCAT Java DTFJ processing status : Started 2010/10/22 17:55:03 MVS Dump Data Set Name. . . : n/a MVS Dump Status . . . . . : Not found  *** Bottom of data. F1=Help      F3=Exit      F5=RptFind    F6=Actions    F7=Up          F8=Down F10=Left     F11=Right    F12=retrieve	

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 between 0 and 99 (both inclusive), then fault entry expiration control is active. The specified value is the number of days that the fault entry will remain locked, following its creation. The time of the day is not taken into consideration, so 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 will not be performed against the fault entry:

- IDIUTIL DELETE processing (for details, see “DELETE control statement” on page 355).

**Note:** The default is to not delete the fault entry, but this can be overridden by using a IDIUTIL Delete user exit (for details, see “IDIUTIL Delete user exit” on page 413).

- 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 76).
- 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 will not prevent deletion of the fault entry.

Any printable character may be entered for this field:

- If a non-printable character is entered, then it will be changed to a '/'.
- If a lowercase character is entered, then it will be 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 512).

The current lock status is shown following the lock flag, and is updated if pressing the Enter key after changing its value.

### 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 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 selected by placing the cursor on it, and pressing Enter. This will result in

## Viewing fault entry information

the Fault Entry Duplicate History display being shown (for details, see “Viewing the fault entry duplicate history” on page 83).

### 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 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 may 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 will not suppress the dump, even if the analysis is deemed complete.)

When exiting from the Fault Entry Information display after having made changes to either of the user-managed fields, a prompt is displayed to confirm that you wish to update the fault entry with the new information. An example of the prompt follows:



```

File  View  Services  Help
----- User Fields Update -----

User information has been modified for the current fault entry. Press
Enter to update the fault entry with the current user fields, or press
PF3/PF12 to exit from the Fault Entry Information display without updating
the fault entry.

History file DSN . . : FRED.HIST
Fault ID . . . . . : F00003

F1=Help    F3=Exit    F12=Cancel

POF Offset. . . . . : 41E
Date. . . . . : 2003/09/19
Time. . . . . : 15:36:47
Job ID. . . . . : JOB30460
Job Execution Class . . . : A
Job Type. . . . . : Batch
Job Step Name . . . . . : G0
EXEC Program Name . . . . : IDISCB1
User ID . . . . . : SWILKEN
System Name . . . . . : MVS2
F1=Help    F3=Exit    F5=RptFind  F6=Actions  F7=Up      F8=Down
F10=Left   F11=Right   F12=retrieve

```

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 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 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

```
File View Services Help
-----
Fault Entry Duplicate History                                     End of data
Command ==> _____ 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 provided differs:

- NoDup(Normal) duplicates

For NoDup(Normal) duplicates, the information 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.

**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 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 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 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 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 selected for copy with the C or CC commands is presented. Included are also any fault entries 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 specified will be 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 will be shown (for details, see “New history file allocation” on page 53).

The original fault ID will be 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 selected for move with the M or MM commands is presented. Included are also any fault entries 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 86 for an example of the Specify Move/Copy Options display that is shown after pressing the Enter key.

The destination history file specified will be 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 will be shown (for details, see “New history file allocation” on page 53).

The original fault ID will be 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 78.

The TSO user's access authorization level required to move a fault entry is ALTER.

---

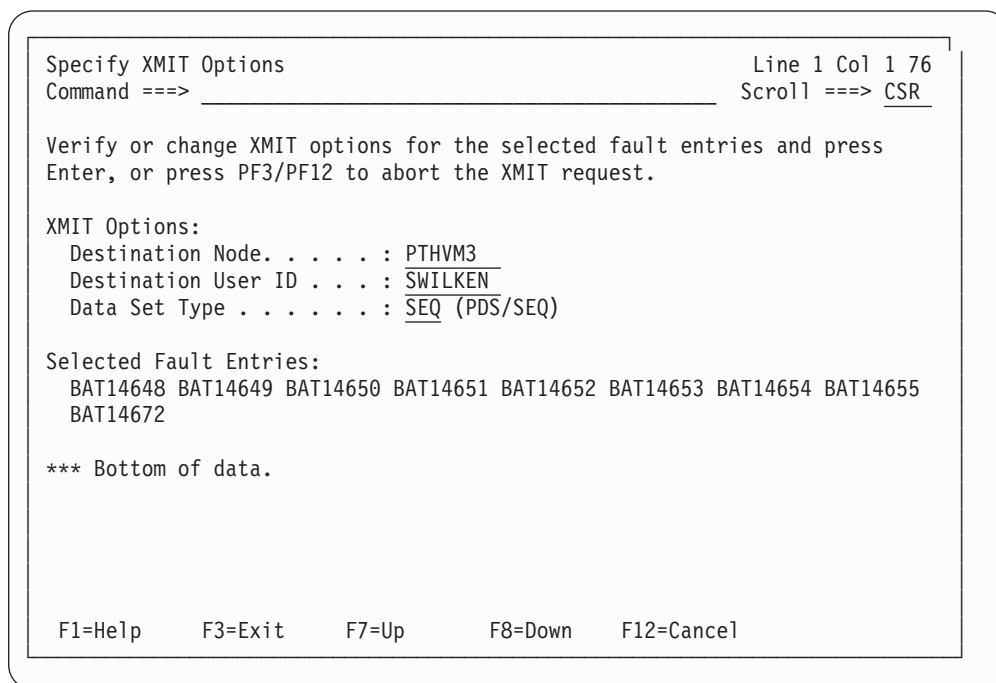
## Transmitting history file entries

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 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). This 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.



The image shows a sample terminal display for the 'Specify XMIT Options' command. At the top, it says 'Specify XMIT Options' with a line number 'Line 1' and column number 'Col 1 76'. Below this, it says 'Command ==>' followed by a blank line. To the right, it says 'Scroll ==>' followed by 'CSR'. The main text says 'Verify or change XMIT options for the selected fault entries and press Enter, or press PF3/PF12 to abort the XMIT request.' Below this, it says 'XMIT Options:' followed by three lines: 'Destination Node. . . . . : PTHVM3', 'Destination User ID . . . . : SWILKEN', and 'Data Set Type . . . . . : SEQ (PDS/SEQ)'. Below this, it says 'Selected Fault Entries:' followed by two lines of fault entry IDs: 'BAT14648 BAT14649 BAT14650 BAT14651 BAT14652 BAT14653 BAT14654 BAT14655' and 'BAT14672'. Below this, it says '\*\*\* Bottom of data.' At the bottom, it says 'F1=Help F3=Exit F7=Up F8=Down F12=Cancel'.

Figure 33. Sample Specify XMIT Options display

Each selected fault entry will be 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 261.

Alternatively, the use of Views (see "Using views" on page 35) in conjunction with data set profile security might be considered.

---

## Chapter 4. Performing batch reanalysis

Batch reanalysis might be used when a user wants to either re-run 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 183.

---

### Batch reanalysis options

To specify general 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 58). This will bring up the Batch Reanalysis Options display as shown in Figure 34 on page 90.

## 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. . : <u>N</u> (Y/N)	
Display panel to edit generated JCL . . . . . : <u>N</u> (Y/N)	
Job card style. . . . . : <u>P</u> (P=Parameters, S=Statements)	
Job Card Parameters:	
Job name suffix . . . . . : <u>A</u> (A-Z, 0-9, @, #, or \$)	
Job class . . . . . : <u>A</u> (A-Z or 0-9)	
Job notify. . . . . : <u>Y</u> (Y/N)	
Job time minutes. . . . . : <u>10</u> (0-99)	
Message class . . . . . : <u>X</u> (A-Z or 0-9)	
Region megabytes. . . . . : <u>0</u> (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. . . . . : <u>N</u> (Y/N)	
Edit the options data set before reanalysis . . . . : <u>N</u> (Y/N)	
*** Bottom of data.	
F1=Help	F3=Exit
F10=Left	F11=Right
F5=RptFind	F6=Actions
F12=Cancel	F7=Up
	F8=Down

Figure 34. Sample Batch Reanalysis Options display

The following may be specified using this display:

### Options line for batch reanalysis

Options that will apply to all batch reanalysis jobs that you submit can be specified here. These options, which will be used in the PARM field of the generated batch reanalysis job, take precedence over any options 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 39 on page 97.

If you need to specify more options than will 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 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 will be shown each time a batch reanalysis is requested, prior to generating the JCL stream.



Having made any changes you wish to make, 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 will not be shown.

### Display panel to edit generated JCL

If this option is set to Y, then you will be presented with an ISPF EDIT display screen of the JCL stream generated by Fault Analyzer as the example shown in Figure 35.

```

File Edit Confirm Menu Utilities Compilers Test Help

EDIT          IBMUSER.SPFTMP1.CNTL                      Columns 00001 00072
Command ==>                                         Scroll ==> PAGE
***** ***** Top of Data *****
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=*
***** ***** Bottom of Data *****

```

Figure 35. Sample batch reanalysis JCL stream EDIT

Having made any changes you wish to make, 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.

See also “Data sets used for batch reanalysis” on page 94 for additional information about the use of this option.

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 90 will follow the General Options section after the Enter key is pressed.
- If S is specified, then a Job Card Statements section will follow the General Options section after the Enter key is pressed.

This would alter the display from that shown in Figure 34 on page 90 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 is the character that will be appended to your user ID to form the job name used for the generated batch reanalysis JCL stream. The default is A.

### Job class

This is the job class that will be 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 will be added to the generated JOB card. If it is set to N, then no NOTIFY parameter will be added. The default is Y.

### Job time minutes

This is the number of minutes that will be used on the TIME parameter of the JOB card. The valid range is from 1 to 30. The default is 10.

### Message class

This is the message class that will be used on the MSGCLASS parameter of the generated JOB card. The default is X.

**Region megabytes**

This is the value that will be used on the REGION parameter of the generated JOB card. The valid range is from 0 to 2047. The default is 0.

**Accounting info**

Anything specified in this field will be 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 will be used as the IDIOPTS user options file. This data set can, for example, be used if more options than will fit on the options line at the top of this display are required.

**Notes:**

1. The options data set will only be used if the "Use this data set during reanalysis" option is set to Y.
2. Options specified on the options line take precedence over options specified in this data set.

**Options member name**

This 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 will be used by Fault Analyzer during the batch reanalysis. If it is set to N, then the data set and member name will not be 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 94.

## Batch reanalysis options

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          MY.OPTIONS(SAMPCNF) - 01.02          Columns 00001 00072
Command ==> _____ Scroll ==> PAGE
***** ***** Top of Data *****
000001 detail(1)
***** ***** Bottom of Data *****

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

Having made your changes to the options data set (if any), 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 89.

---

## Data sets used for batch reanalysis

When performing batch reanalysis through the ISPF interface, the generated JCL will include DD statements as required for any JOBLIB, STEPLIB, or Fault Analyzer compiler listing or side file data sets. DD statements will be 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 specified via the IDIOPTS user options file or the PARM field will 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 89), 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.COBO
//
// DD DISP=SHR,DSN=DA.LISTING.COBO
/* 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 may 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.COBO='LIST,MAP,Source,XREF'
//COBO.SYSIN DD DISP=SHR,DSN=MY.COBO.Source(MYPGMA)
//COBO.SYSPRINT DD DISP=SHR,DSN=&SYSUID..COBLIST(MYPGMA)
/*
/* Recompile MYPGMB
/*
//CBLCOMP EXEC IGYWC,PARM.COBO='LIST,MAP,Source,XREF'
//COBO.SYSIN DD DISP=SHR,DSN=MY.COBO.Source(MYPGMB)
//COBO.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 will enable identification of the line of source code where the error occurred.

Note that only one program is compiled in each job step. This 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 310.

## Creating your own batch reanalysis job

You can optionally add a 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 16.

Any options 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 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 194.

---

### Interactive reanalysis options

To specify general interactive reanalysis options that apply to your interactive reanalysis sessions only, first select Interactive 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 58). This will bring up the Interactive Reanalysis Options display as shown in Figure 39.

```
File View Services Help
Interactive Reanalysis Options                               Line 1 Col 1 80
Command ==> _____ Scroll ==> CSR

Press PF3 to save options or PF12 to cancel.

General Options:
Options line for
interactive reanalysis. . . : LA(JPN)
Redisplay this panel
before each reanalysis. . . : N (Y/N)
Display panel to alter
allocated data sets . . . : N (Y/N)
Prompt before opening a
SYSMDUMP. . . . . : Y (Y/N)
Prompt for missing side
files . . . . . : Y (Y/N)

Reanalysis Options Data Set Control:
Options data set name . . : JCLLIB
Options member name . . : IDICNF00 (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.
```

Figure 39. Sample Interactive Reanalysis Options display

## Interactive reanalysis options

The following may be specified using this display:

### Options line for interactive reanalysis

Options that will apply to all interactive reanalysis sessions that you initiate can be specified here. These options, which are the equivalent of the PARM field options used by batch reanalysis jobs, take precedence over any options specified through an options file (see “Options data set name” below).

The option LA(JPN) is shown as an example on the options line in Figure 39 on page 97.

### Redisplay this panel before each reanalysis

If this option is set to Y, then the Interactive Reanalysis Options display will be shown each time an interactive reanalysis is requested.

Having made any changes you wish to make, 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 alter allocated data sets” option below.

If this option is set to N, then the Interactive Reanalysis Options display will not be shown.

### Display panel to alter allocated data sets

If this option is set to Y, then you will be presented with an ISPF EDIT display screen of the pseudo JCL stream generated by Fault Analyzer as the example shown in Figure 40.

```
File Edit Confirm Menu Utilities Compilers Test Help
EDIT      IBMUSER.SPFTMP1.CNTL                      Columns 00001 00072
Command ==> _____ Scroll ==> 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

Having made any changes you wish to make in accordance with the instructions displayed, then enter the EXIT (PF3) or CANCEL (PF12) command as appropriate to initiate the interactive reanalysis.

See also “Data sets used for interactive reanalysis” on page 160 for additional information about the use of this option.



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

If this field is set to Y and, during the interactive reanalysis or as a result of displaying storage locations from within the interactive report, access is required to a storage location that is not contained in the saved minidump, a display is shown before opening an associated SYSMDUMP or SVC dump data set to look for the missing storage.

An example of this display is shown in Figure 41.

File Options View Services Help									
Confirm SYSMDUMP Open _____									
Command ==> _____									
Fault Analyzer has determined the need to open SYSMDUMP data set: JKATNIC.CICS53.SOS.DUMP									
Permitting this might cause delays, however, if the open is not permitted, Fault Analyzer cannot access important storage information.									
Press Enter to confirm the data set open.									
Press Cancel or Exit to cancel the data set open and attempt to continue without access to missing storage locations.									
F1=Help      F3=Exit      F12=Cancel									
	F00286	CICS53	n/a	MVS2	S122	2001/05/22	10:49:44		
	F00325	DAAMB022	n/a	MVS2	S0C6	2001/04/27	11:03:48		
	F00111	CICS53	n/a	MVS2	S08E	2001/03/22	13:12:23		
	F00272	CICS53	n/a	MVS2	S08E	2001/03/22	13:12:23		
i	F00328	CICS53	n/a	MVS2	S08E	2001/03/22	13:12:23		
F1=Help      F3=Exit      F4=MatchCSR      F5=RptFind      F6=Actions      F7=Up									
F8=Down      F10=Left      F11=Right      F12=MatchALL									

Figure 41. Sample Confirm SYSMDUMP Open display

You will only be prompted at most once during any interactive reanalysis session. If the open is canceled by entering CANCEL or EXIT, no further attempts will be made to open the SYSMDUMP data set. Likewise, if the open is confirmed, Fault Analyzer will check 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 will be opened if required without prompting you first.

#### Prompt for missing side files

A single character (Y or N) that controls whether or not the Compiler Listing Not Found display is shown when no compiler listing or side file could be 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

## Interactive reanalysis options

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 will be 'N'. Otherwise, it will be 'Y'.

By clearing this field, and pressing Enter, the value will be re-initialized to its default setting.

### 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 will be used as the IDIOPTS user options file. This data set can, for example, be used if more options than will fit on the options line at the top of this display are required.

#### Notes:

1. The options data set will only be used if the “Use this data set during reanalysis” option is set to Y.
2. Options specified on the options line take precedence over options specified in this data set.

### Options member name

This 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 will be used by Fault Analyzer during the interactive reanalysis. If it is set to N, then the data set and member name will not be 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 101.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      USER.JCLLIB(IDIOPTS) - 01.02      Columns 00001 00072
Command ==> _____ Scroll ==> PAGE
***** ***** Top of Data *****
000001 datasets(idiexec(user.exec))
000002 exits(listing(rexx(1x)))
***** ***** Bottom of Data *****

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

Having made your changes to the options data set (if any), enter the EXIT command (usually mapped to PF3).

## 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.

## Initiating interactive reanalysis

```

File  Options  View  Services  Help
-----
IBM Fault Analyzer - Fault Entry List                               Line 1 Col 1 80
Command ==>                                                         Scroll ==> CSR

Fault History File or View  : 'SWILKEN.HIST'

{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).}

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
S
S      _____ Interactive Reanalysis Status _____
S
S      Current Function      : le_heap
S      Total Elapsed Time    : 11 Seconds
S      Total CPU Time        : 8.201 Seconds
S
F1=H      F1=Help      F2=Split      F3=Exit      F9=Swap      F12=Cancel
F8=D

```

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 103 shows an example of the first interactive report display from where all other parts of the interactive report can be selected:

```

File View Services Help
-----
Interactive Reanalysis Report                               Line 1 Col 1 80
Command ==>                                         Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7                      MVS2    2001/12/21 13:02:25

Fault Summary:
Module IDISCBL1, program IDISCBL1, 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 72), then information about the maximum amount of allocated storage that Fault Analyzer used during the analysis will be included at the bottom of the display. This amount of storage is for explicit allocations only and does not include storage for loaded modules, etc.

The interactive report will be 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 will format the report accordingly. This is true also if the screen size is dynamically resized—just press the Enter key and the report section viewed will be 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 tabable 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 will take you to other parts of the report while others will 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 additional information and an example of this display, refer to “Displaying source code” on page 135.

### Storage address

Displays the storage at this location in both hexadecimal and translated EBCDIC. For additional information and an example of this display, refer to “Displaying storage locations” on page 138.

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 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.

The image shows a terminal window titled 'Interactive Reanalysis Report'. At the top is a menu bar with 'File View Services Help'. Below the title bar, it shows 'Command ===>' and 'Line 1 Col 1 80'. The main display area shows 'JOBNAME: IDIVPCOB SYSTEM ABEND: 0C7 MVS2 2001/12/21 13:02:25'. A confirmation box is displayed with the text: 'Are you sure you want to exit the interactive report ? Press Enter to confirm exit or press PF12 to return to the interactive report.' Below this box, it says 'F1=Help F12=Cancel'. At the bottom of the terminal, there is a list of options: '3. Open Files', '4. Storage Areas', '5. Messages', '6. Language Environment Heap Analysis', '7. Abend Job Information', '8. User Notes', and '9. Fault Analyzer Options'. Below the list, it says '{Fault Analyzer maximum storage allocated: 1.47 megabytes.}' and '\*\*\* Bottom of data.'

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.

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 478.

An additional prompt might be displayed if user information has been modified. For details, see “Refresh processing” on page 161.

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.

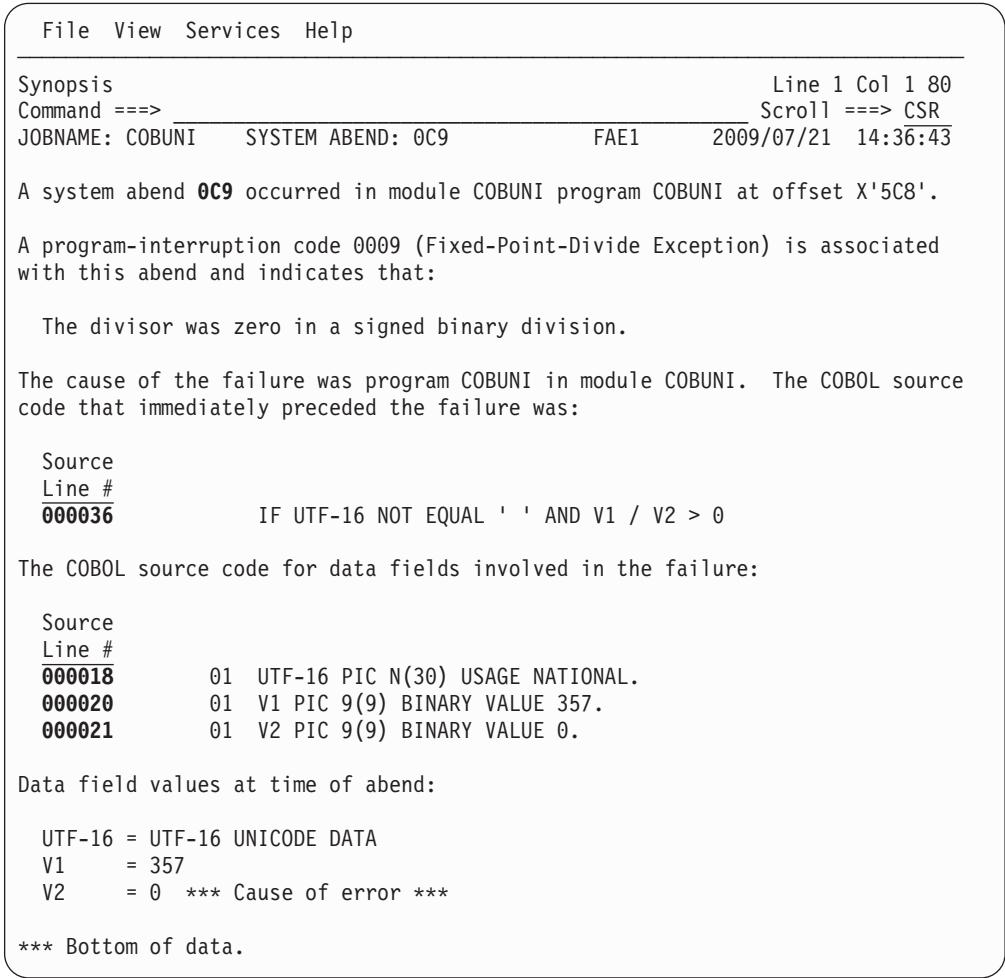


Figure 46. Sample Synopsis display

## 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 106.

## Primary option: Event Summary

File View Services Help							
Event Summary					Line 1 Col 1 80		
Command ==>					Scroll ==> CSR		
TRANID: FRED		CICS ABEND: AEIL			2002/05/24 13:49:18		
{The following events are presented in chronological order.}							
Event #	Type	Fail Point	Module Name	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	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	
F8=Down		F10=Left		F11=Right		F6=Actions	
F7=Up							

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 107.

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 138.
- 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 135.

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 will



disappear, but will reappear again if the display is scrolled to the top. For general information about help text, see “Adding or removing help text” on page 72.

The column headings will never be scrolled out of view. However, if scrolling horizontally, the column headings will 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 108.

## 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.G0SET.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(S)
                             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)

```

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

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:
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 48. Sample Event Details display for point of failure (Part 2 of 2)

**Note:** The COBOL PERFORM traceback shown at **1** might only appear if the code has not been inlined by the compiler. This will be the case for optimized programs and non-optimized programs which have only one PERFORM of a paragraph or section.

If using an IDILANGX file, then it will need to have been created using a version of Fault Analyzer at the PTF UK48699 level (August 2009), or later.

## Primary option: Event Summary

All information 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 134) and associated storage areas (refer to “Displaying associated storage areas” on page 130).

To select the previous or next event from the one currently displayed, point-and-shoot links in yellow are provided at the bottom of the current display. Simply place the cursor on one of these and press the Enter key.

---

## Primary option: Open Files

Selecting the “Open Files” option will provide 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 49.

```
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 49. 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 50 on page 111.

(You return from the Open Files display by entering the Exit (PF3) command.)

```

File View Services Help

File Information
Command ==>
JOBNAME: COBTSTC  SYSTEM ABEND: 0C9  FAE1  2007/09/18  09:02:09
Line 1 Col 1 80
Scroll ==> CSR

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
16599FC0  C9C9CF9 F9F94040 40404040 40404040 *III999 *
16599FD0  +10  40404040 40404040 * *

Last Record Written -1. . : RDW=00240300 Segment data length 32, variable re
Address  Offset  Hex  EBCDIC
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
16599EF0  40404040 40404040 40404040 40404040 * *
16599F00  +10  40404040 40404040 * *

Current Record Build Area : RDW is zero, no length assigned yet
Address  Offset  Hex  EBCDIC
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 50. Sample File Information display

If the data set shown at **1** is QSAM, VSAM, or IAM, and exists, then it will be presented as a point-and-shoot field, that if selected, will show a selection menu from where either Edit, View or Browse can be chosen. If IBM File Manager for z/OS is available, then it will be 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 additional information about the presentation of open file buffers, see "Open file record information" on page 183.

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 51 on page 112.

## Primary option: CICS Information

File View Services Help

Associated File Control Blocks

Line 1 Col 1 80

Command ==> Scroll ==> CSR

JOBNAME: COBTSTC SYSTEM ABEND: 0C9 FAE1 2007/09/18 09:02:09

Data Extent Block (DEB) at Address 008BCD84 :

Address	Offset	Hex	EBCDIC
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
0001A038		1656C858 00000000	*..H.....*
0001A040	+8	00FF0000 0EF15026 002FBE96 07022FE8	*.....1&....o...Y*
0001A050	+18	00004000 00006B70 E6000001 5801E8B4	*.. ...,W....Y.*
0001A060	+28	007C0048 008BCD84 92C9D870 00C8B348	*.@.....dkIQ..H..*

Figure 51. Sample Associated File Control Blocks display

## Primary option: CICS Information

Selecting the “CICS Information” option will provide you with a display of information related to CICS, as the example shown in Figure 52.

File View Services Help					
CICS Information				Line 1 Col 1 80	
Command ==>				Scroll ==> CSR	
TRANID: CS31		CICS ABEND: ASRA		FAB2	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					
3. Last CICS 3270 Screen Buffer					
4. Last CICS 3270 Screen Buffer Hex					
5. Summarized CICS Trace					
6. CICS Trace Formatting					
7. CICS Recovery Manager					
8. CICS Levels, Commareas, and Channels					
*** Bottom of data.					

Figure 52. Sample CICS Information display

The CICS Information display provides options, that can be entered on the command line or selected using the cursor, to additional CICS information, such as the following.

## Last CICS 3270 Screen Buffer

Selecting the “Last CICS 3270 Screen Buffer” link will provide you with a display of the last 3270 screen buffer written by CICS as the example shown in Figure 53.

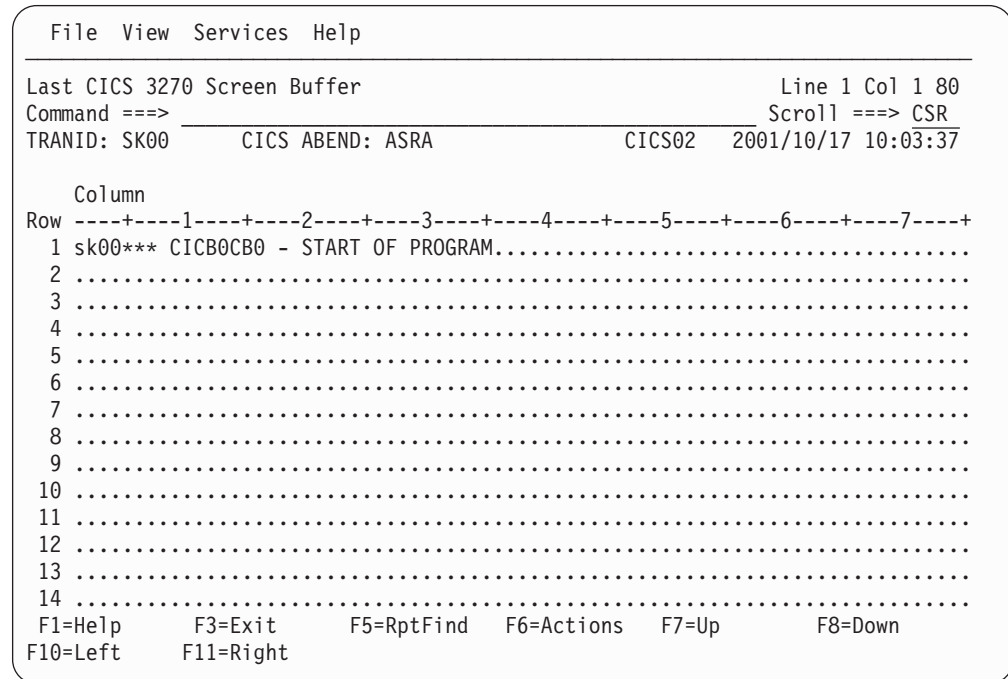


Figure 53. Sample Last CICS 3270 Screen Buffer display

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 will be 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 will provide 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 54 on page 114.

### Primary option: CICS Information

[illegible]

Figure 54. 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 113.

## Summarized CICS Trace

Selecting the “Summarized CICS Trace” link will provide you with a display of the CICS trace table as the example shown in Figure 55 on page 115.



File View Services Help									
Summarized CICS Trace								Line 1 Col 1 80	
Command ==>								Scroll ==>	CSR
TRANID: CS31			CICS ABEND: ASRA			FAB2		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_DUMP/EXCEPTION_DUMP/NOT				
00027 QR	DU	0601	DUTM	EXIT	LOCATE_SYSTEM_DUMP/OK			FFFFFFFF,1,0,0,N0,YE	
00027 QR	DU	0601	DUTM	EXIT	COMMIT_SYSTEM_DUMP/OK				

Figure 55. 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 can be selected by placing the cursor on them and pressing Enter. This will provide a full source listing display as shown in “Displaying source code” on page 135.

## CICS Trace Formatting

Selecting the “CICS Trace Formatting” link will provide you with a display that permits specific selection parameters to be entered for the CICS trace, as the example shown in Figure 56 on page 116.

## Primary option: CICS Information

File View Services Help

S  
C  
C  
T

CICS Trace Selection Parameters

Specify CICS trace selection parameters and press Enter.

Format . . . . . A (Abbrev/Short/Full)

Exception Only . . N (Yes/No)

A Sequence Start . . 000001

C End . . . 000375

C Highlight Interval 0.128 (0-99.999999999 secs)

C Task IDs . . . . . 00027

C KE Task Numbers

Terminal IDs . . .

S Transaction IDs

Time Start . . . . (HHMMSS)

End . . . . . (HHMMSS)

Domain/Point IDs

ine 1 Col 1 80  
roll ==> CSR  
5/22 14:22:19

7. CICS Levels, Commareas, and Channels

\*\*\* Bottom of data.

Figure 56. Sample CICS Trace Selection Parameters display

The following parameters can be specified:

### Format

Specifies the level of trace formatting: A indicates the abbreviated, one line per entry, form of the trace; S indicates short formatting consisting of the abbreviated entry plus fully formatted parameter list, return address, time and interval; F indicates you want a 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 any number up to 5 digits in length; any of JAS, J01 through J99, III, TCP or DST; or a 2 character domain ID of the attaching domain (for non-TCA tasks).

### KE Task Numbers

Specifies up to five 4 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 lower case set Caps to N and enter all IDs as case-sensitive. When Caps is Y all entries are translated to upper-case.

**Transaction IDs**

Specifies the transaction identifiers of up to five transactions for which trace entries are to be displayed. If any transaction IDs contain lower case set Caps to N and enter all IDs as case-sensitive. When Caps is Y all entries are translated to upper-case.

**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 five 2 character 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), optionally followed by a trace point ID within that domain. The trace point ID may be up to 4 hexadecimal digits in length (if less than 4 digits are entered, it is treated as a generic value).

Having made any optional changes to parameters, press Enter to view the CICS trace. An example of the CICS Trace display is shown in Figure 57.

File View Services Help									
CICS Trace							Entry 1 Col 1 80		
Command ==>							Scroll ==>	CSR	
Abbrev	Short	Full	TRANID: CS65			IDISNAP CALL	FAE2		
00027	QR	SM 0D02	SMMF	EXIT	FREEMAIN/OK		CICS storage at 174E7748		2
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 57. 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 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 will provide you with a display as the example shown in Figure 58 on page 118.

## Primary option: CICS Information

```
File View Services Help
CICS Levels, Commareas, and Channels
Command ==>
TRANID: COBA      CICS ABEND: ATCV      FAE1      2005/04/06 12:32:44
Line 1 Col 1 80
Scroll ==> CSR

Number of Link Levels . . . : 5

Level 1 of 5 : Load Module DFHPIDSH

Fault Analyzer Event #. . . : n/a

Load Module Address . . . . : 178E7F90
Entry Point Address . . . . : 178E7FB8
Addressing Mode . . . . . : AMODE 31
Current Channel Name. . . . : DFHAHC-V1
Count of Containers . . . . : 18

1
Container DFHWS-OPERATION at address 16A4A490 has a length of X'11'
Data Offset X'000000' EBCDIC CICSVER20operation

Container DFHWS-BODY at address 16A4A440 has a length of X'3FE'
Data Offset X'000000' ASCII <SOAP-ENV:Body>

2
Container DFHWS-SOAPLEVEL at address 16A4A3A0 has a length of X'4'
Data Offset X'000000' HEX 00000001
```

Figure 58. Sample CICS Levels, Commareas, and Channels display

Selecting the container name point-and-shoot field provides information about the container type. For example, selecting the DFHWS-OPERATION point-and-shoot field at **1**:

```
File View Services Help
CICS Levels, Commareas, and Channels
Command ==>
TRANID: COBA      CICS ABEND: ATCV      FAE1      2005/04/06 12:32:44
Line 1 Col 1 80
Scroll ==> CSR

DFHWS-OPERATION is a container of DATATYPE(CHAR) that is normally used in a
service provider application deployed with the CICS Web services assistant. It
holds the name of the operation that is specified in a SOAP request.

*** Bottom of data.
```

Figure 59. Sample CICS container type display

Selecting the "data" point-and-shoot field provides information about the data type. For example, selecting the "data" point-and-shoot field at **2**:

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.				
Data Length X'3FE' <b>3</b> <u>Format XML Data</u>				
Address	Offset	Hex	ASCII	
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 60. Sample container data 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 **3** shows the formatted XML data:

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 61. 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 62.

```
File View Services Help
-----
System-Wide Messages
Command ==>
JOBNAME: P00398      SYSTEM ABEND: 0CB      MVS2      2003/03/24 13:54:05
Line 1 Col 1 80
Scroll ==> CSR

Event 5 Messages

CEE3211S Job-specific text not available

Non-Event-Related Messages

$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 62. Sample System-Wide Messages display

## Primary option: DB2 Information

Selecting the “DB2 Information” option will provide you with a display of information related to DB2 as the example shown in Figure 63 on page 121.

```

File View Services Help
DB2 Information
Command ==>
JOBNAME: DAC2DH07  SYSTEM ABEND: 806          MVS2      2002/11/26 09:44:50
Line 1 Col 1 80
Scroll ==> CSR

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 63. 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 HVAR5
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
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 63. Sample DB2 Information display (Part 2 of 3)



## DB2 Control Blocks

SQL Communications Area (SQLCA) for Event # 7 Program DAPNDH11 at Address **1747E9F8** :

Offset	Field	Value	EBCDIC	
Dec	Hex	Name	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	FFFFFFFF92 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 F8=Down  
F10=Left F11=Right

Figure 63. 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 will be shown in this display.

## Primary option: IMS Information

Selecting the “IMS Information” option will provide you with a display of information related to IMS as the example shown in Figure 64.

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
IMS Version . . . . . : V7R1M0			
IMS Region Type . . . . . : Online Control Region (DB/DC)			
IMS Subsystem Name. . . . . : IB21			
Application Program Name. . : ICCB0010			
PSB Name. . . . . : DFHSAM25			

Figure 64. 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 . . . . . : 1730A3D0
  DL/I Call Function. . . : GU (Get Unique)

Parameter 2 . . . . . : 00016398
  (See "IMS Control Blocks" for details of this PCB)

Parameter 3 . . . . . : 1730A310

Parameter 4 . . . . . : 9730A3B0
  SSA # 1 . . . . . : PARTROOT(      =      )

IMS Control Blocks

Input/Output Program Communications Block (IOPCB):
  At Address. . . . . : 00016320
  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 Block (DBPCB) (** Current/Last Used **):
  At Address. . . . . : 00016398
  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
  000163F4          F0F2C1D5 F9F6F0C3 F1F04040 40404040 *02AN960C10 *
  00016404      +10 40                                *          *
```

```
JCB DL/I Call Trace (Most recent call first):
  Call      Status
  # Code Description      Code Description
  1 01  GHU or GU          ' '      Status good.
```

Figure 64. Sample IMS Information display (Part 2 of 3)

```

IMS Accounting Information

DL/I Data Base Calls:
GU Calls. . . . . : 1
GN Calls. . . . . : 0
GNP Calls . . . . . : 0
GHU Calls . . . . . : 0
GHN Calls . . . . . : 0
GHNP Calls. . . . . : 0
ISRT Calls. . . . . : 0
DLET Calls. . . . . : 0
REPL Calls. . . . . : 0
Total Calls . . . . . : 1
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

IMS Parameter Modules

Module DFSPRPX0 Address . . : 000079C0

*** Bottom of data.
F1=Help      F3=Exit      F5=RptFind    F6=Actions    F7=Up      F8=Down
F10=Left     F11=Right

```

Figure 64. Sample IMS Information display (Part 3 of 3)

The IMS Information display provides:

- General information about the IMS region
- Last DL/I call parameter list
- Detailed information for 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.

### Primary option: Storage Areas

Selecting the “Storage Areas” option will provide 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 65.

```
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 65. Sample System-Wide Storage Areas display

Selecting any event-related point-and-shoot links from this display (such as the "Event 4 Program COBMAIN Storage Areas" link above), by either entering the option number on the command line or by placing the cursor on the option number point-and-shoot field and pressing Enter, results in a display which 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 130 for additional 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 66 on page 127.

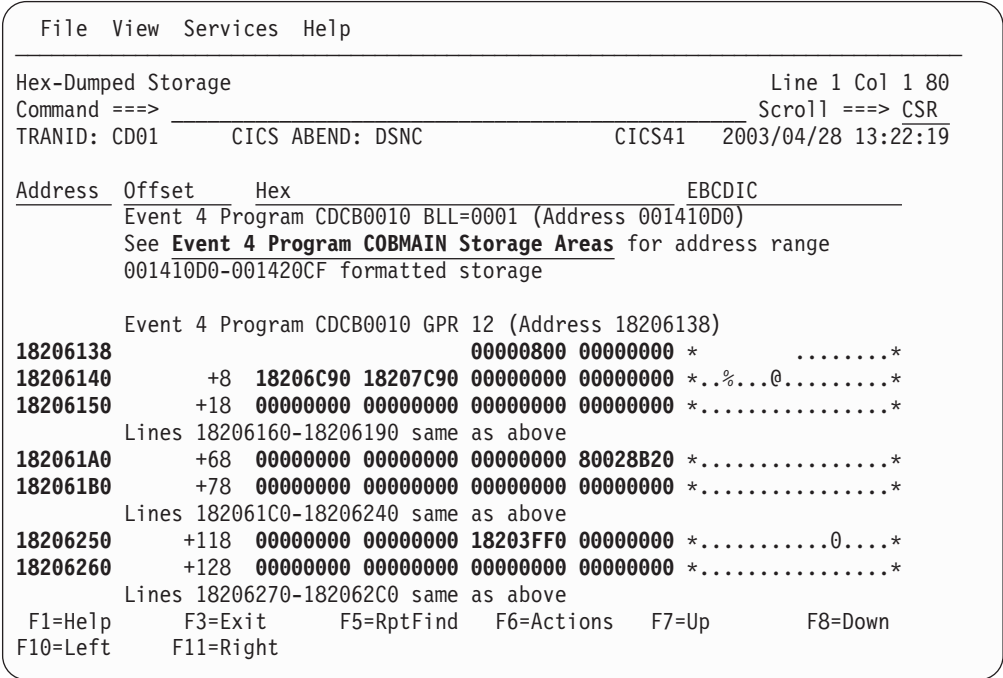


Figure 66. Sample Hex-Dumped Storage display

Primary option: Java Information

See Chapter 8, “Performing Java analysis,” on page 175.

Primary option: WebSphere Information

See Chapter 8, “Performing Java analysis,” on page 175.

Primary option: Language Environment Heap Analysis

Selecting the “Language Environment Heap Analysis” option will provide you with a display of information related to the LE heap as the example shown in Figure 67 on page 128.

## 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	F AE2	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 67. Sample Language Environment Heap Analysis display

## Primary option: MTRACE Records

Selecting the “MTRACE Records” option will provide you with a display showing the MVS master trace, as the example shown in Figure 68.

File View Services Help					
MTRACE Records				Line 1 Col 1 80	
Command ==>				Scroll ==>	CSR
SLIP DUMP ID=F092				F AE1	2006/08/21 18:11:11
M 0100000	F AE1	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	F AE1	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	F AE1	06233	18:10:40.63	JOB08017	00000090 IEF677I WARNING MESSAG
N 0020000	F AE1	06233	18:10:40.67	JOB08017	00000090 ICH70001I ANDYMEL LAS
N 4000000	F AE1	06233	18:10:40.68	JOB08017	00000090 \$HASP373 DACBB012 STAR
N 0000000	F AE1	06233	18:10:40.71	JOB08017	00000090 IEF403I DACBB012 - STA
N 0004000	F AE1	06233	18:10:40.99	JOB08017	00000290 -
-PAGING COUNTS---					
N 0004000	F AE1	06233	18:10:40.99	JOB08017	00000290 -JOBNAME STEPNAME PRO
AGE SWAP VIO SWAPS STEPNO					
N 0004000	F AE1	06233	18:10:41.01	JOB08017	00000290 -DACBB012 V00
0	0	0	0	1	
N 0004000	F AE1	06233	18:10:41.25	JOB08017	00000290 -DACBB012 D00

Figure 68. 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.

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 69.

File View Services Help		Line 1 Col 1 80
Abend Job Information		Scroll ==> CSR
Command ==>		
TRANID: FRED	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 69. Sample Abend Job Information display

This display provides information about the environment that existed when the fault was analyzed in real-time. The information 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 67), and results in the User Note List display being shown (see page 144).

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 70 on page 130.

## Displaying associated storage areas

```
File View Services Help
-----
Fault Analyzer Options                                     Line 1 Col 1 80
Command ==>                                             Scroll ==> CSR
TRANID: FRED          CICS ABEND: AEIL                  CICS04   2001/09/25 11:06:56

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.}

FaultID(F00066)
Language(ENU)
NoLocale

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 DA.SYSADATA
```

Figure 70. 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 has been made 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 with a compiler listing or side file 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 71 on page 131.



File View Services Help			
Associated 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 <b>17096448</b> for length 344			
WORKING-STORAGE SECTION			
- Collapse hex - Collapse level 88			
Off Hex Value	Data Value	Source (Starting a	
BLW=0000 at address <b>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 71. Sample Associated Storage Areas display

By scrolling down through the displayed information, you will also find Linkage Section information, File Section information, etc., as appropriate for the current program.

Two features are unique to the interactive reanalysis report: The ability to hide the hex-value column and the ability to collapse level 88 items.

## 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 71 and pressing Enter, the display is changed to that shown in Figure 72 on page 132:

## 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.
300 1.111100e+02          05 ELEMENT-1                            COMP-1.
304 *....                * 05 ELEMENT-2 OCCURS 4 TIMES          PIC XXXX
                               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 72. 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 71 on page 131 and pressing Enter, the display is changed to that shown in Figure 73 on page 133:

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 <b>17096448</b> for length 344			
WORKING-STORAGE SECTION			
- Collapse hex + Expand level 88			
Off Hex Value	Data Value	Source (Starting a	
BLW=0000 at address <b>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-
<b>Level 88 Items</b>			
			ELEMENT-
314 00000000	*....	*	05 ELEMENT-
318 00000000	0		05 ELEMENT-

Figure 73. 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 74 on page 134.

## Displaying associated storage areas

```
File View Services Help
Level 88 Items
Command ==>
JOBNAME: COBLVL88  SYSTEM ABEND: 0CB          FAE1      2011/06/24  14:22:45
Line 1 Col 1 80
Scroll ==> CSR

Source
      88 JACK VALUE 'JACK'.
      88 JILL VALUE 'JILL'.

*** Bottom of data.
```

Figure 74. Sample level 88 items display

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.

---

## Expanding messages and abend codes

Messages or abend codes are initially never expanded when using the interactive dump reanalysis feature of Fault Analyzer. This 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 will bring up a display similar to what you see in the batch report as the example shown in Figure 75 on page 135.

```

File View Services Help
-----
Message Explanation                                     Line 1 Col 1 80
Command ==>                                           Scroll ==> CSR
JOBNAME: DACBB045  USER ABEND: 4038                    MVS2    2001/10/10 10:38:22

IGZ0035S There was an unsuccessful OPEN or CLOSE of file INDD in program
          DAVSA009 at relative location X'0380'. Neither FILE STATUS nor an
          ERROR declarative were specified. The status code was 39. 1

IGZ0035S
IGZ0035S There was an unsuccessful OPEN or CLOSE of file file-name in
          program program-name at relative location location. Neither FILE
          STATUS nor an ERROR declarative were specified. The status code was
          status-code.

Explanation: An error has occurred while opening or closing the named
file. No file status or user error declarative was specified.

Programmer Response: Check to make sure there is a DDname defined for the
indicated file.

F1=Help      F3=Exit      F5=RptFind   F6=Actions   F7=Up        F8=Down
F10=Left     F11=Right

```

Figure 75. 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” will replace 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 could not be found, then the text “explanation not available” will replace 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 shown in Figure 76 on page 136 might be displayed.

## Displaying source code

```

File View Services Help
-----
Program CICS FRED 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'0602f00007000008000f0f0f2
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'04043000070000000100004000
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 76. Sample Compiler Listing display

Note that the source line or statement number initially selected is highlighted.

The example shown here assumes that the display of pseudo assembler instructions is suppressed—for details on this, see “Displaying source code” on page 135.

Information displayed in blue (all lines starting in column 1) pertains to the program source code. On the left side of the display is information about the source line and/or listing statement number of the source displayed. This 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 58). This will cause the Compiler Listing display to be reformatted with pseudo assembler instructions inserted into the program source code as shown in Figure 77 on page 137.

```

File View Services Help
-----
Program CICS FRED Compiler Listing                               Line 316 Col 1 80
Command ==>                                                    Scroll ==> CSR
TRANID: FRED          CICS ABEND: AEIL          CICS04 2002/01/09 15:37:46

000003DA 1845          LR      R4,R5
000003DC 8E40 0020      SRDA   R4,32
000003E0 5D40 C000      D      R4,0(,R12)
000003E4 4040 8008      STH    R4,8(,R8)
000066      *EXEC CICS READ FILE('NOTTHERE') RIDFLD(RID)
000067      *      INTO(REC-AREA) END-EXEC.
000068      Move length of REC-AREA to dfhb0020
000003E8 D201 8088 A01C MVC   136(2,R8),28(R10)
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.

000003EE D210 D0B8 A061 MVC   184(17,R13),97(R10)
000003F4 4130 D0B8      LA     R3,184(,R13)
000003F8 5030 D0A0      ST     R3,160(,R13)
000003FC D207 D0D0 A08E MVC   208(8,R13),142(R10)
00000402 4130 D0D0      LA     R3,208(,R13)
00000406 5030 D0A4      ST     R3,164(,R13)

F1=Help    F3=Exit    F5=RptFind  F6=Actions  F7=Up      F8=Down
F10=Left   F11=Right

```

Figure 77. Sample Compiler Listing display: Pseudo assembler instructions enabled

Information displayed in green (all lines starting in column 15) pertains to the disassembly of machine instructions. These are 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, you might first want to ensure that the source line of interest is scrolled to the top of the display prior to adding pseudo assembler instructions to the listing display, as the additional display lines might otherwise 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.

If you scroll to the top of the listing, for example by issuing the UP MAX command, you will see information about from where the compiler listing or side file was obtained. An example is shown in Figure 78 on page 138.

## Displaying storage locations

```
Options Help
-----
Program CICFRED Compiler Listing                               Line 1 Col 1 80
Command ==>                                                    Scroll ==> CSR
TRANID: FRED          CICS ABEND: AEIL          CICS04    2002/01/09 15:37:46

The listing or side file used for the following was found in
DA.LISTING.COBOL(CICFRED).

Source
Line #
000001      *****
000002      * TRANSACTION: FRED                                *
000003      *   EXPECTED OUTPUT:                                *
000004      *   'STARTED CICFRED' FOLLOWED BY AEIL ABEND        *
000005      *****
000006      IDENTIFICATION DIVISION.
000007      PROGRAM-ID. CICFRED.
000008      ENVIRONMENT DIVISION.
000009      DATA DIVISION.
000010      WORKING-STORAGE SECTION.
000011      77 UPPER-LIMIT PIC S9(4) COMP VALUE 255.
F1=Help    F3=Exit    F5=RptFind  F6=Actions  F7=Up      F8=Down
F10=Left   F11=Right
```

Figure 78. 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 69 for details).

An example of the storage display is shown in Figure 79.

```
File View Services Help
-----
Dump Storage                               17C01380-17C013D7
Command ==>                               Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7      MVS2    2003/08/12 13:46:58

Address  Offset  Hex                                EBCDIC
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 5108D203 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.....*
17C013D0      +50  00000000 00000000                                *.....*
Module IDISCBL1 CSECT CEEBETBL
F1=Help    F3=Exit    F7=Up      F8=Down  F10=Prev  F11=Next
```

Figure 79. Sample Dump Storage display



Placing the cursor anywhere in the hexadecimal storage display area, and pressing the Enter key, will take you to the selected address. If the point-and-shoot field in which the cursor is placed is less than 8 digits, then it will automatically be 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, will ensure that the address is interpreted as a 24-bit address.

A 64-bit address is selected by overtyping the last digit of the point-and-shoot field which represent the first half of the 64-bit address (bits 0-31), or by overtyping 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 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 overtyping the last digit of the first field like this

```
0000000_ 80109020
```

or the first digit of the second field like this

```
00000001 _0109020
```

will result in the 64-bit address 00000001\_80109020 being displayed. As with 31-bit addresses, the second half of the 64-bit address will automatically be padded with leading zeroes to 8-digits.

Given the following

A record is maintained of the last 10 addresses displayed. You may use the PREV (PF10) or NEXT (PF11) commands to re-display areas previously selected.

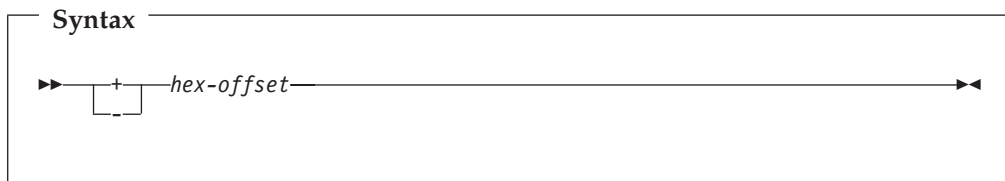
The number of bytes per line shown depends on the visible width, not the current preferred formatting width. 32 bytes per line will be shown if the visible width permits, otherwise 16 bytes will be shown.

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 140).

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 65.

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:

## Displaying storage locations



where *hex-offset* is a hexadecimal offset relative to the top left address shown. For example:

+10C

or

-D4

## 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 will apply, typing one or more characters that are distinguishable from hexadecimal digits, and pressing Enter. For example, given the Dump Storage display shown in Figure 79 on page 138, placing the cursor at the address 17C01389 storage, and typing "This is", the following display would be expected:

```

File View Services Help
-----
Dump Storage                                     17C01380-17C013D7
Command ==>                                     Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7             MVS2      2003/08/12 13:46:58

Address  Offset      Hex                                EBCDIC
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:

```

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 will be the only line of the note shown if the display of the note is later collapsed.

Assuming that the final note is as follows:

## 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 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 will show the newly created user note, inserted immediately ahead of the storage to which it belongs.

```
File View Services Help
-----
Dump Storage                      17C01380-17C013D7
Command ==>                      Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7          MVS2      2003/08/12 13:46:58

Address  Offset  Hex                                EBCDIC
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 51          *...-/.&-.*
- 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 will be "collapsed" as shown below:

```

File View Services Help
-----
Dump Storage                                     17C01380-17C013D7
Command ==>                                     Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7             MVS2      2003/08/12 13:46:58

Address  Offset      Hex                                EBCDIC
Module IDISCB1 program IDISCB1 + X'640', source line # 32
Event 1 Program IDISCB1 GPR 1 + X'1051'
Event 1 Program IDISCB1 GPR 10 + X'1300'
Event 1 Program IDISCB1 GPR 11 + X'1008'
Event 1 Program IDISCB1 GPR 12 + X'1348'
Event 1 Program IDISCB1 GPR 14 + X'680'
Event 1 Program IDISCB1 GPR 3 + X'600'
Event 1 Program IDISCB1 GPR 4 + X'1544'
17C01380      20004160 61345060 51                *...-/.&-.*
+ 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 IDISCB1 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 additional 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. This will result in a display like the example shown in Figure 80 on page 144:

## Displaying storage locations

```
File View Services Help
-----
User Note List                                     Line 1 Col 1 80
Command ==>                                     Scroll ==> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              MVS2    2003/08/12 13:46:58

- Collapse all / + Expand all
{The following line commands are available: E (Edit), D (Delete), S (Show), +
 (Expand), - (Collapse). To enter a line command, overtype the +/- sign in
 column 1, or simply place the cursor on the +/- sign and press Enter to
 perform the default expand/collapse action indicated.}

      Address  Text
- 17C01389 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 fa
- 17C01610 So is this!

F1=Help      F3=Exit      F5=RptFind   F6=Actions   F7=Up        F8=Down
F10=Left     F11=Right
```

Figure 80. 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 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 81 on page 145:

```

File View Services Help
User Notes Update
-----
I
C User notes have been added or modified for the current fault entry. Press
J Enter to update the fault entry with the current user notes, or press
F PF3/PF12 to exit from the interactive report without updating the fault
M entry.
S History file DSN . : 'IDI.HIST'
i Fault ID . . . . . : F00264
F1=Help F3=Exit F12=Cancel

4. Storage Areas
5. Messages
6. Language Environment Heap Analysis
7. Abend Job Information
8. Fault Analyzer Options

{Fault Analyzer maximum storage allocated: 2.61 megabytes.}

*** Bottom of data.
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down
F10=Left F11=Right

```

Figure 81. Sample User Notes Update prompt

Pressing Enter from this display will permit the history file fault entry to proceed, while pressing PF3 or PF12 will cause exit 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 62 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 will be shown a popup window similar to the following:

## Mapping storage areas using DSECT information

File View Services Help

Storage DSECT Mapping

Enter the name of the Dsect in the Dsect Name field to be used to map the storage address provided in the Address field. Press PF4 to display a list of all available Dsects. Optionally a specific Dsect can be used by supplying a Dataset and Member name in the DSN field. In this case if a Dsect name is not provided it will be made equal to the member name.

Address \_\_\_\_\_  
Dsect Name \_\_\_\_\_  
DSN . . . \_\_\_\_\_

F1=Help F3=Exit F12=Cancel

First Operand Address . . : 0002A120 (3808 bytes of storage addressable)  
First Operand Length. . . : 8  
First Operand Storage . . : 00000000 0986888C \*.....fh.\*  
Second Operand Address. . : 0002A110 (3824 bytes of storage addressable)  
Second Operand Length . . : 4  
Second Operand Storage. . : C1C2C3CF \*ABC.\*  
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
F10=Left F11=Right

Figure 82. 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 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 147 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 will be mapped and displayed, similar to the following example:



File View Services Help									
Dsect mapping for DFHCSADS at address 4de20								Line 1 Col 1 80	
Command ==>								Scroll ==>	CSR
CICS DUMP: SYSTEM=QXPM2C61 CODE=ASRA ID= MVS2 2003/06/25 13:47:55									
0004DE20 +0000				DSECT DFHCSADS					
0004DE20 +0000				DFHCSABA EQU *					
				00000248 0000D0A0 17EB4D00 983C1ECE					
				80BF4DA8 80800000 18685160 18642330					
				000003FD 18973FB8 00000BAF 00000000					
				983C1A40 18973000 18685160 18684B70					
				00051D80 17F90680					
				CSAOSRSA DS 18F					
				CSASOSI DS 0B					
0004DE68 +0072 00				CSASSI1 DS B					
				CSAFPURG EQU X'80'					
				CSAFTCAB EQU X'40'					
				CSASDTRN EQU X'20'					
				CSACSDOP EQU X'02'					
				CSASOSON EQU X'01'					
				CSAKCMI DS 0B					
0004DE69 +0073 10				CSASSI2 DS B					
F1=Help		F3=Exit		F5=RptFind		F6=Actions		F7=Up	
F10=Left		F11=Right						F8=Down	

Figure 83. Sample Storage DSECT Mapping Map display

Scroll up/down or right/left as needed to display additional 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.

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)”), 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, and hence, if new DSECTs are added to a data set in the IDIDSECT concatenation, or if the \$DINDEX member is updated, then it will be necessary to either restart the interactive report, or 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)” on page 148) should contain a line for every DSECT in

## Mapping storage areas using DSECT information

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)” on page 147 for details about this member). It does this by 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 will need to call the macro in question, providing the required keywords, and will 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 will not detect the TCTENIB DSECT, as this 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 will be 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 will create 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 69, 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 will be shown a popup panel similar to the following:

File View Services Help
Storage RUNCHAIN Command

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 00000000, FFFFFFFF)  
Eyecatcher Text . . . . .  
Eyecatcher Offset . . . . . (Hex)

F1=Help F3=Exit F12=Cancel

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 84. Sample Storage RUNCHAIN Command entry display

For a given Start Address and Forward Pointer Offset, the RUNCHAIN command will follow 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:
  - X'00000000'
  - X'FFFFFFFF'
  - The initial start address—implying the chain has looped
  - A user supplied End Of Chain Identifier
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:

## Displaying chained data areas

FileViewServicesHelp

Storage RUNCHAIN Command

Enter the required fields and press Enter .

Start Address . . . . . 0005C000

Max Number Control Blocks 9999 (Decimal)

Forward Pointer Offset . . . 1C (Hex)

End of Chain Identifier . . . (Hex, Default Values 00000000, FFFFFFFF)

Eyecatcher Text . . . . . DFHSMQPH

Eyecatcher Offset . . . . . 2 (Hex)

F1=Help

F3=Exit

F12=Cancel

000011n/a01PARM1PIC X(4) .

000012n/a01PARM2.

Data Field Values:

PARM1 = 0001

PARM2 = HEADING FOR IDIXSNAP

F1=HelpF3=ExitF5=RptFindF6=ActionsF7=UpF8=Down

F10=LeftF11=Right

Pressing Enter, the following display is presented:

FileViewServicesHelp

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

AddressHex (First 32 Bytes Of Data)

0005C00000306EC4 C6C8E2D4 D8D7C840 40404040 C1D76DE3 C3C1F2F4 002AA000 0005

0005B00000306EC4 C6C8E2D4 D8D7C840 40404040 C1D76DE3 C3C1F2F4 0005C000 0005

0005A00000306EC4 C6C8E2D4 D8D7C840 40404040 C1D76DE3 C3C1F2F4 0005B000 0005

0005900000306EC4 C6C8E2D4 D8D7C840 40404040 C1D76DE3 C3C1F2F4 0005A000 2E80

2E80CCCCInvalid eyecatcher. Expected: >DFHSMQPH

Found : 24.....Q.

2E80CCCC C3C1F2F4 2E80CC14 1ED82728 C0010100 00000000 00000600 00059000 002A

002AA00000306EC4 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 62), 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 will be shown a popup panel similar to

the following:

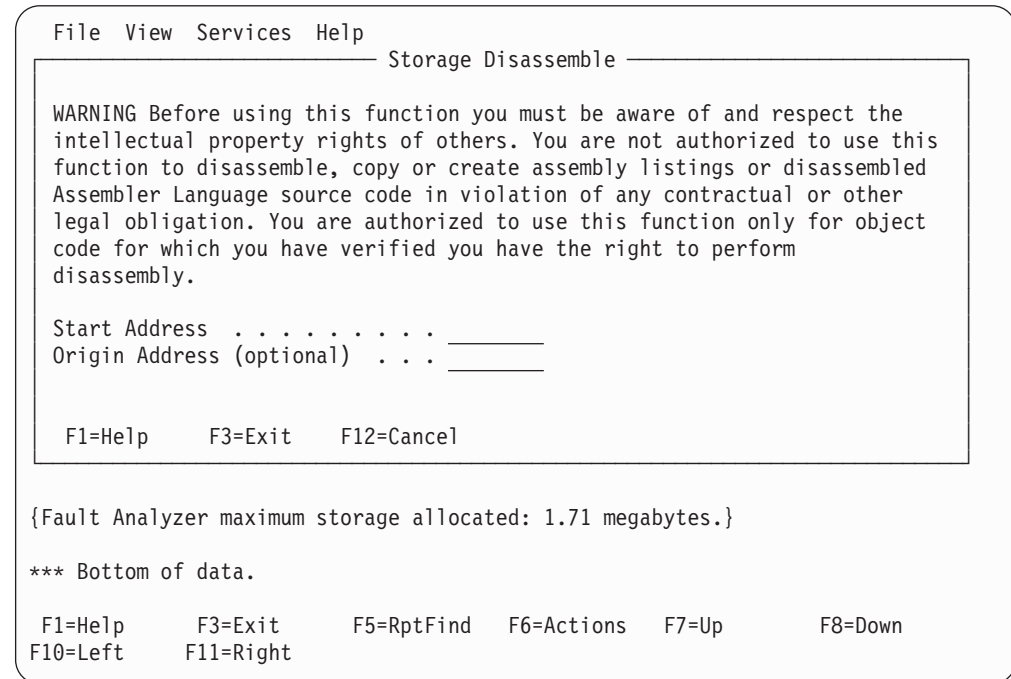
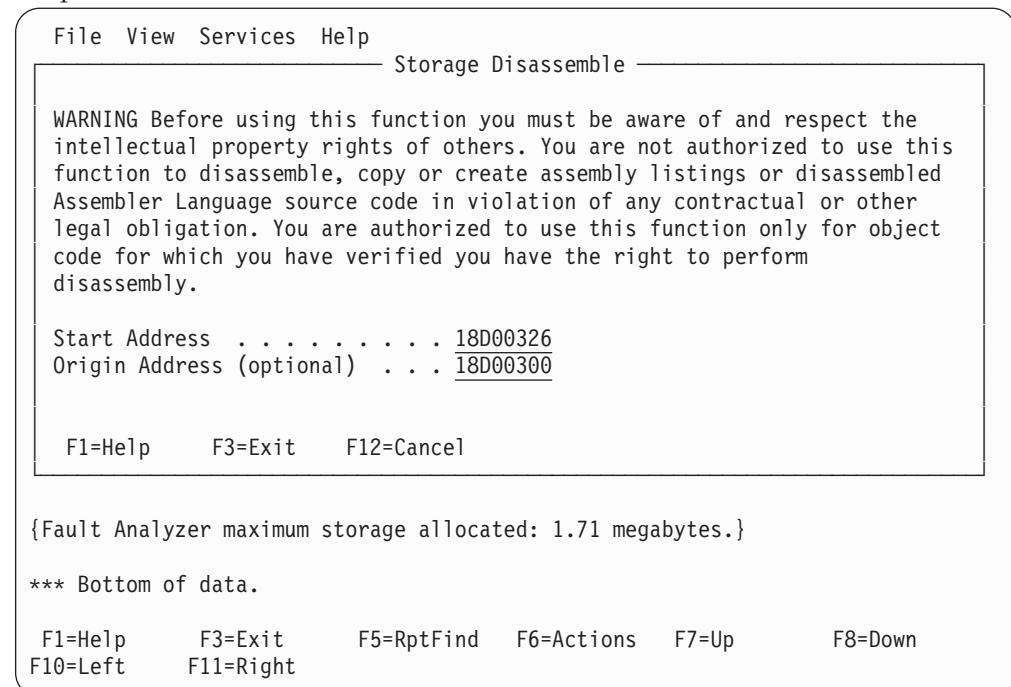


Figure 85. Sample Storage Disassemble display

The DISASM command will attempt 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:



## Disassembling object code

Pressing Enter, the following display is presented:

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 70), 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 will be shown a popup panel similar to the following:

File View Services Help
STCK Conversion

Enter the 16 hex character STORE CLOCK (STCK) value in the field and press ENTER to display its Date Time value.

STCK Value  
Date Time :

F1=Help F3=Exit F12=Cancel

Most recently referenced data items:  
Data Item . . . . . : BLW=0000+D6B  
At Address. . . . . : 0009ADF3  
Length. . . . . : X'2'  
Data Item Storage . . . : 4040 \* \*  
Data Item . . . . . : BLW=0002+23F  
At Address. . . . . : 0009C2C7  
Length. . . . . : X'4'  
Data Item Storage . . . : 40404040 \* \*  
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
F10=Left F11=Right

Figure 86. 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:

File View Services Help
STCK Conversion

Enter the 16 hex character STORE CLOCK (STCK) value in the field and press ENTER to display its Date Time value.

STCK Value B99F67D5 FBD00DC0  
Date Time :

F1=Help F3=Exit F12=Cancel

Most recently referenced data items:  
Data Item . . . . . : BLW=0000+D6B  
At Address. . . . . : 0009ADF3  
Length. . . . . : X'2'  
Data Item Storage . . . : 4040 \* \*  
Data Item . . . . . : BLW=0002+23F  
At Address. . . . . : 0009C2C7  
Length. . . . . : X'4'  
Data Item Storage . . . : 40404040 \* \*  
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down  
F10=Left F11=Right

Pressing Enter, the display is updated as follows:

## Converting STORE CLOCK values

FileViewServicesHelp

STCK Conversion

Enter the 16 hex character STORE CLOCK (STCK) value in the field and press ENTER to display its Date Time value.

STCK Value B99F67D5 FBD00DC0  
Date Time : 2003/06/25 05:44:48

F1=HelpF3=ExitF12=Cancel

Most recently referenced data items:  
Data Item . . . . . : BLW=0000+D6B  
At Address. . . . . : 0009ADF3  
Length. . . . . : X'2'  
Data Item Storage . . . : 4040 \* \*  
Data Item . . . . . : BLW=0002+23F  
At Address. . . . . : 0009C2C7  
Length. . . . . : X'4'  
Data Item Storage . . . : 40404040 \* \*  
F1=HelpF3=ExitF5=RptFindF6=ActionsF7=UpF8=Down  
F10=LeftF11=Right

To exit from the STCK command, enter EXIT (PF3).

---

## User-specific report formatting

Using the EXEC command (see “EXEC” on page 63), 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 390.

Non-REXX Formatting user exits can not 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:



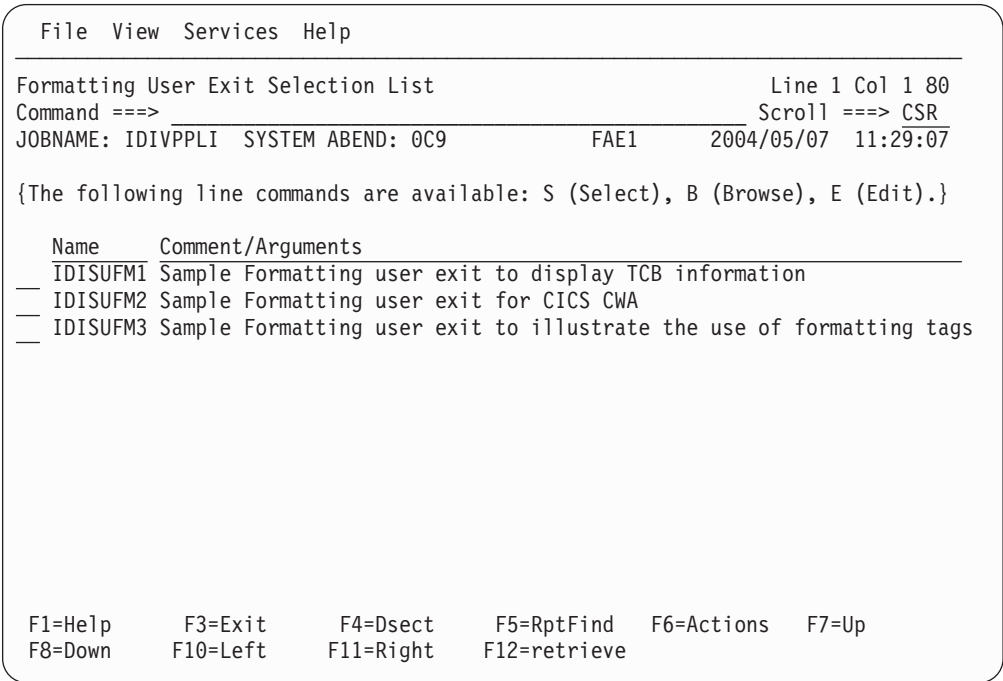


Figure 87. Sample Formatting User Exit Selection List display

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:

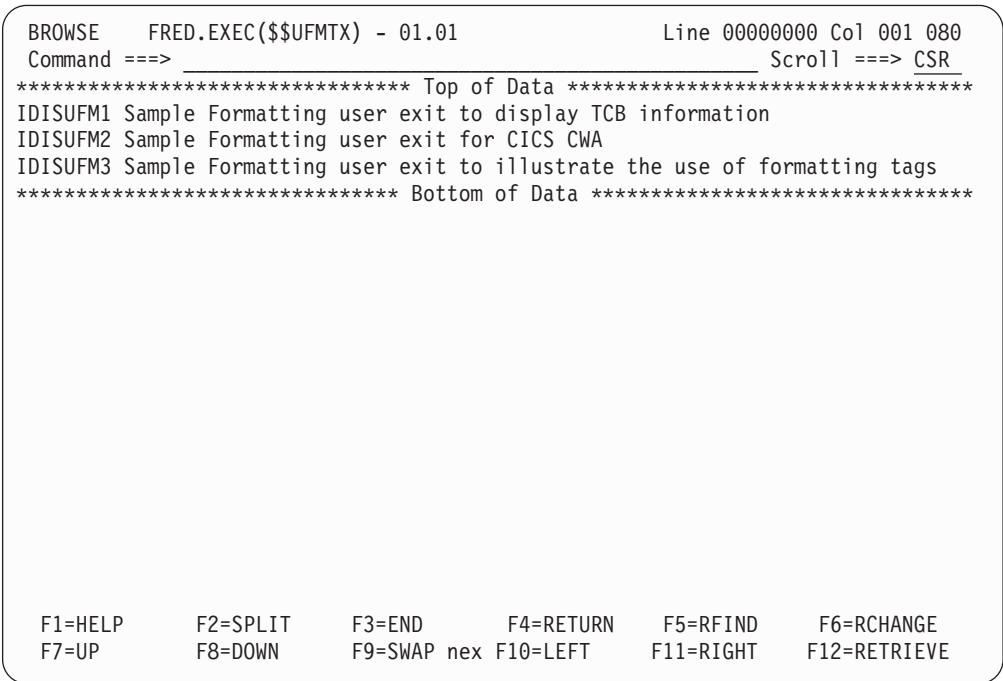


Figure 88. Sample \$\$UFMTX member

## User-specific report formatting

The above sample control member is provided in softcopy format as member IDISUFMX in data set IDI.SIDISAM1.

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 should present a display similar to that in Figure 87 on page 155 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. This 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. You will notice a color change of the field when it has been overtyped to indicate that the data will be 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 89 on page 157, will be 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 Fault Analyzer IDILANGX side file, a COBOL SYSDEBUG side file generated by using the TEST(NONE,SYM,SEPARATE) compiler option, or an Enterprise PL/I side file generated by using the TEST(STMT,SYM,NOHOOK,SEPARATE) compiler option.

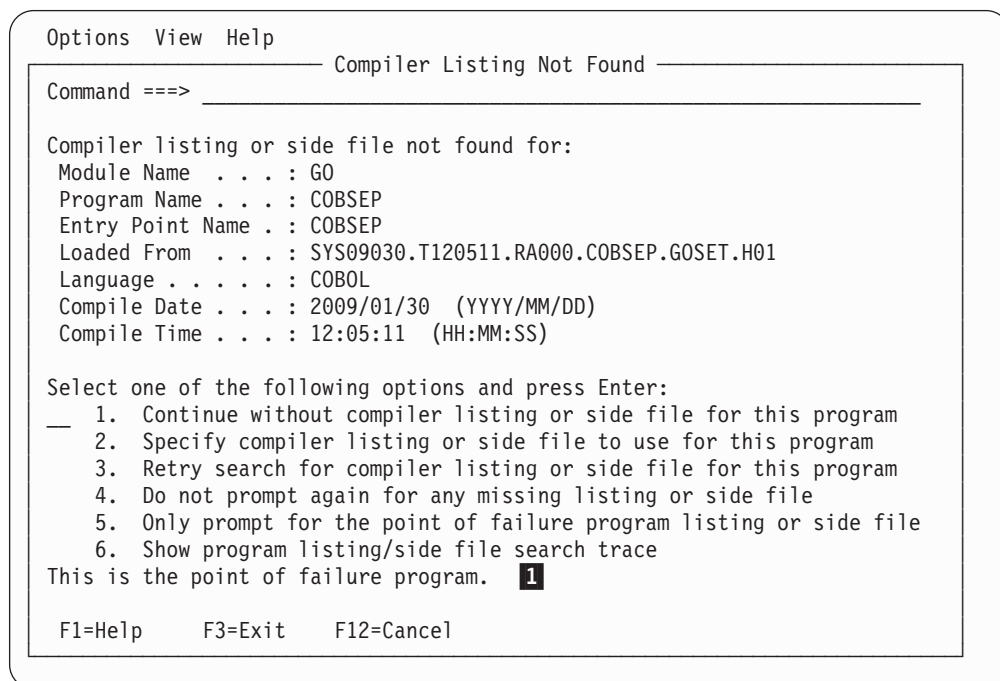


Figure 89. Sample Compiler Listing Not Found display

The prompt provides you with these choices:

## 1. Continue without compiler listing or side file for this program

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 this program

This option will display 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 that should be used for the current program as the example shown in Figure 90 on page 158.

## Prompting for compiler listing or side file

File Options View Services Help

Compiler Listing Not Found

Command ==>

Specify Compiler Listing or Side File

Command ==>

Specify the data set and member name containing the compiler listing or side file and press Enter.

Data Set Name . . .

Member . . . . . IDISCBL1

3. Retry search for compiler listing or side file for this program

4. Do not prompt again for any missing listing or side file

5. Only prompt for the point of failure program listing or side file

6. Show program listing/side file search trace

This is the point of failure program.

BAT16771 SWILKEN 136 Yes IDIDUMP.IDIRFR.FAE1.D090116.T01545

BAT16768 SWILKEN 139 Yes IDIDUMP.IDIRFR.FAE1.D090116.T01462

Figure 90. 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 data set name specified is stored in the ISPF profile for your application ID and is used for initialization of the display.

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 will need to change this field.

If a sequential data set is specified, then the member name will be ignored.

Having specified 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 an additional prompt as the example shown in Figure 91 on page 159 is presented.

```

Listing/Side File Mismatch
Line 1 Col 1 76
Command ==> _____ Scroll ==> CSR
Listing/Side File . . . . . : JERRYBL.SYSDEBUG.COBOL(COBSEP)
Compile Date/Time:
  Load Module . . . . . : 2009/01/30 12:05:11
  Listing/Side File . . . . : 2009/01/30 12:05:58
Program COBSEP has a COBOL SYSDEBUG file signature or checksum mismatch.
The number of DATA DIVISION STATEMENTS is 11 in the side file, 10 in the
load module. The number of PROCEDURE DIVISION STATEMENTS is 15 in the side
file, 13 in the load module.
NOTE: If the compile mismatch is significant, and the file is accepted,
      then some information presented might not correctly reflect the
      conditions at the time of the fault.
Press ENTER to continue with this side file, or F3/F12 to cancel.
*** Bottom of data.
F1=Help    F3=Exit    F5=RptFind F7=Up      F8=Down    F12=Cancel

```

Figure 91. 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 will be presented, for example, data fields with incorrect values, incorrect source lines or statements, etc.

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 91 will instead show the display in Figure 89 on page 157 again, which allows for a different side file to be provided.

### 3. Retry search for compiler listing or side file for this program

Selecting this option will cause 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 will not be 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 will 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 will only prompt 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** ).

## Prompting for compiler listing or side file

### 6. Show program listing/side file search trace

If you select this option, then a trace of the search for a matching compiler listing or side file, as the example shown in Figure 92, is presented.

```
Listing/Side File Trace
Listing/Side File search trace for COBSEP          Line 1 Col 1 76
Command ==>                                     Scroll ==> CSR

LE compile date 2009-01-30 time 12:05:11
JERRYBL.SYSDEBUG.COBOL SYSDEBUG side file
  Rejected - Member not found or COBOL SYSDEBUG signature check failed
DA.WDBLANGX(COBSEP)
  Rejected - Member not found
JERRYBL.IDILANGX(COBSEP)
  Rejected - Member not found
DA.LISTING.COBOL(COBSEP)
  Rejected - Member not found
JERRYBL.LISTING.COBOL(COBSEP)
  Rejected - Timestamp later than load module, WORKING-STORAGE length
    check failed
    - Timestamp date 2009-02-03 time 12.00.39
    - Listing WORKING-STORAGE length x'D2' load module
      WORKING-STORAGE length x'CA'

*** Bottom of data.

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

Figure 92. Sample Listing/Side File Trace display

This trace is equivalent to what can be obtained when using the IDITRACE DDname, as shown in “IDITRACE information” on page 313.

When the Compiler Listing Not Found display on page 157 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 will occur or not. For details about this option, see “Interactive reanalysis options” on page 97.

---

## Data sets used for interactive reanalysis

When performing interactive reanalysis through the ISPF interface, pre-allocation will be performed as required for any Fault Analyzer compiler listing or side file data sets that were used in real-time. Allocations will be 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 specified via the IDIOPTS user options file or the PARM field will 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 97), 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 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 93 is presented upon exiting the interactive report, which permits the cancellation of the refresh, or the suppression of the minidump.

Fault Entry Refresh
Line 1 Col 1 76

Command ==> \_\_\_\_\_
Scroll ==> CSR

User information has been added or modified.

Press Enter to refresh the current history file entry, or press PF3/PF12 to cancel.

A minidump does not currently exist for this fault entry, but one will be saved unless suppressed using the option below.

MaxMinidumpPages Option . . : 100  
 Current Minidump Pages. . . : 163  
 Suppress Minidump . . . . : N (Y/N)

\*\*\* Bottom of data.

Figure 93. 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 493).





---

## Chapter 6. Performing CICS system abend dump analysis

A feature unique to the interactive component of Fault Analyzer is the ability to analyze information 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 97.

#### 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 377
- “Compiler Listing Read user exit” on page 382
- “Message and Abend Code Explanation user exit” on page 386
- “Formatting user exit” on page 390

---

### 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 58). This brings up the Analyze MVS Dump Data Set display as shown in Figure 94 on page 164.

## Selecting a CICS dump data set

File View Services Help

Analyze MVS Dump Data Set

Line 1 Col 1 80

Command ==> Scroll ==> CSR

Enter the name of a MVS SVC or SYSMDUMP data set and press Enter to initiate a performing analysis, issue the Exit (PF3) or Cancel (PF12) command.

Dump Data Set Name. . . . : 'cics.dump1'

\*\*\* Bottom of data.

*Figure 94. 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 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 95 on page 165).

```

File Options View Services Help
IBM Fault Analyzer - Fault Entry List
Command ==>
Line 1 Col 1 80
Scroll ==> CSR

Fault History File or View : 'SWILKEN.DEMO.HIST'

{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).}

Fault ID Job/Tran User ID Sys/Job Abend Date Time
---
F00323 IDIVPCOB SWILKEN MVS2 S0C7 2001/12/21 13:02:25
F00445 ALLANT01 JACKIED MVS8 S0C7 2001/12/19 03:29:57
F00442 ALLANT01 ALLANT MVS8 S0C7 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 S0C4 2001/08/02 17:03:18
F00035 CICS53 n/a MVS2 n/a 2001/04/05 14:49:11
F00034 CI
F00294 DB
Analyzing MVS dump data set. Please wait...
F1=Help F7=Up
F8=Down F10=Left F11=Right F12=MatchALL

```

Figure 95. 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 96, from which a selection can be made by typing an S against the desired address space, and pressing Enter.

```

File Options View Services Help
Address Space Selection
Row 1 to 2 of 2
Command ==>
Please use S to select the ASID to analyze.
ASID Jobname Job Type
_ X'004F' CICS5ANS CICS
_ X'008A' IDISS
***** Bottom of data *****
F1=Help F3=Exit F7=Up F8=Down F12=Cancel

SW00009 ICCB0010 SWILKEN MVS2 S0CB 2003/11/26 14:29:18
SW00008 ICCB0010 SWILKEN MVS2 S0CB 2003/11/25 14:07:15
SW00007 ICCB0010 SWILKEN MVS2 S0CB 2003/11/25 09:29:18
SW00006 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 10:51:16
SW00005 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 10:46:00
SW00004 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 10:35:29
SW00003 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 10:19:27
SW00002 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 08:46:35
SW00001 ICCB0010 SWILKEN MVS2 S0CB 2003/11/21 08:33:53
F1=Help F3=Exit F4=MatchCSR F5=RptFind F6=Actions F7=Up
F8=Down F10=Left F11=Right F12=MatchALL

```

Figure 96. 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 will be 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 97).

```
File View Services Help
CICS System Abend Interactive Reanalysis Report
Command ==>
CICS DUMP: SYSTEM=CICSDI CODE=SM0002 ID= D381 2002/08/13 09:55:03
Line 1 Col 1 80
Scroll ==> CSR

Select one of the following options and press Enter to access further fault
information:
1. Synopsis
2. Abend Job Information
3. CICS System Information
4. Options in Effect

DFHSM0002 CICSDI A severe error (code X'030E') has occurred in module DFHSMGF.

Severity 3 Observations

*** Bottom of data.

F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down
F10=Left F11=Right
```

Figure 97. 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 desired option. 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 10 or more severity 2 observations are available, then a point-and-shoot link that will take you to a separate display is provided instead.

- 3 Problems that generally would not be considered contributing reasons for the fault, but merely items of interest.  
  
If 5 or more severity 3 observations are available, then a point-and-shoot link that will take 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 100 on page 169).

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 98.

FileViewServicesHelp

Synopsis

Line 1 Col 1 80

Command ==>

Scroll ==> CSR

CICS DUMP: SYSTEM=CICSDI

CODE=SM0002

ID=

D381

2002/08/13 09:55:03

Fault Information:

CICS Product Level . . . . . : V5 R3 M0

Dump ID. . . . . : 4/0007

Dump Code. . . . . : SM0002

Date/Time. . . . . : 2002/08/13 09:55:03 (Local)

Message. . . . . : DFHSM0002 CICSDI A severe error (code X'030E')

has occurred in module DFHSMGF.

Symptoms . . . . . : PIDS/565501800 LVLS/530 MS/DFHSM0002

RIDS/DFHSMGF PTFS/UQ52744 PRCS/0000030E

Title. . . . . : n/a

Caller Address . . . . . : n/a

\*\*\* Bottom of data.

F1=Help

F3=Exit

F5=RptFind

F6=Actions

F7=Up

F8=Down

F10=Left

F11=Right

Figure 98. 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 99 on page 168.

## Displaying the CICS system abend interactive report

```
File View Services Help
Abend Job Information
Command ==>
CICS DUMP: SYSTEM=CICSDI CODE=SM0002 ID= D381 2002/08/13 09:55:03
Line 1 Col 1 80
Scroll ==> CSR

IBM Fault Analyzer Abend Job information:

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

Execution Environment:

Operating System . . . . . : OS/390 V02R10M00
Data Facility Product. . . : DFSMS OS/390 V2R10M0
CPU Model. . . . . : 2064
F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down
F10=Left F11=Right
```

Figure 99. 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 100 on page 169.

```

File View Services Help
CICS System Information                                     Line 1 Col 1 80
Command ==> _____ Scroll ==> CSR

Select one of the following options and press Enter:

1. CICS Task Summary
2. Error History
3. Storage Usage by Task

AI - AutoInstall Manager      AP - Application Domain
BR - Bridge Information       CC - Catalog Domains
CQ - Console Queue Component  CSA - Common System Area
DB2 - DB2 Information         DD - Directory Domain
DH - Document Handler Domain  DLI - DL/I Information
DM - Domain Manager          DP - Debug Profile Domain
DS - Dispatcher Domain        DU - Dump Domain
EJ - Enterprise Java Domain   FC - File Control
IC - Interval Control         IS - ISC/IP Domain
KE - Kernel Domain           LD - Loader Domain
LG - Log Manager Domain       LM - Lock Manager Domain
ME - Message Domain           MN - Monitoring Domain
MRO - Multiregion Option      NQ - Enqueue Domain
OT - Object Transaction Domain PA - Parameter Domain
PG - Program Manager Domain   PI - Pipeline Manager Domain
PR - Partner Resource Manager PT - Partner Domain
RM - Recovery Manager Domain  RS - Region Status Domain
RZ - Request Stream Domain    SIT - System Initialization Table
SJ - SJ (JVM) Domain          SM - Storage Manager Domain
SO - Sockets Domain           SSA - Static Storage Areas
ST - Statistics Domain        TCP - Terminal Control Definitions
TD - Transient Data Domain    TI - Timer Domain
TMP - Table Manager           TR - Trace Domain
TS - Temporary Storage Domain US - User Domain
UEH - Global User Exit Details WB - Web Domain
XM - Transaction Manager Domain XS - Security Domain

LCK - Lock Owner/Waiter Information
TRC - CICS Trace
NMT - MVS Name/Token Pairs

*** Bottom of data.

```

Figure 100. 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 tabable and shown in reverse highlight mode, then this 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 table rows which 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 which match the specified string will be 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

## Displaying the CICS system abend interactive report

the already sorted table data. Also, once table data has 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 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 101 on page 171.



```

File View Services Help
Options in Effect
Command ==>
CICS DUMP: SYSTEM=CICSDI CODE=SM0002 ID= D381 2002/08/13 09:55:03
Line 1 Col 1 80
Scroll ==> CSR

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
IDIEXEC IBMUSER.EXEC
IDIHIST IDI.HIST
IDILC PROD.LISTING.C
IDILCOB PROD.LISTING.COBOL
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 101. 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 exiting from the CICS System Abend Interactive Reanalysis Report, you are presented with the option to create a history file entry as shown in Figure 102 on page 172.

## Creating a history file entry

File View Services Help

Create History File Entry

To create a history file entry for the analyzed MVS dump data set, specify a history file data set name and press Enter.

History file DSN . . 'SWILKEN.DEMO.HIST'

Minidump pages . . : 29

Suppress minidump . . N (Y/N)

F1=Help F3=Exit F12=Cancel

DFHSM0002 CICSDI A severe error (code X'030E') has occurred in module DFHSMGF.

**Severity 3 Observations**

\*\*\* Bottom of data.

F1=Help F3=Exit F5=RptFind F6=Actions F7=Up F8=Down

F10=Left F11=Right

Figure 102. Sample Create History File Entry display

To create a history file entry, change the name of the history file in which the entry should be created if necessary (using ISPF data set name specification rules), optionally indicate if a minidump with the indicated number of pages should not be created by changing the “Suppress Minidump” option to 'Y', and press Enter. The “history file DSN” field is by default initialized with the current history file name selected on the Fault Entry List display, or, if a view is being used, the first history file name specified in the view member. 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, and a `MATCH FAULT_ID fault_id` command is issued so that only the newly created fault entry is displayed. This is done since the newly created fault entry might not be at the top of the chronologically ordered Fault Entry List display, and therefore might be difficult to locate. To again see all fault entries in the history file, enter the `MATCH ALL` command (usually mapped to PF12).

Creating a fault history entry for a CICS system abend, while permitting the writing of a minidump, improves the performance of subsequent reanalysis of the fault.

---

## 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 58). This brings up the Format CICS Auxiliary Trace Data Set display as shown in Figure 103.

```
File View Services Help

Analyze MVS Dump Data Set                               Line 1 Col 1 80
Command ==>                                             Scroll ==> CSR

Enter the name of a CICS auxiliary trace data set and press Enter to initiate
formatting. To return from this display without formatting trace, issue the
Exit (PF3) or Cancel (PF12) command.

Trace Data Set Name . . . . : 'cics.trace1'

*** Bottom of data.
```

*Figure 103. 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 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 104 on page 174).

## Specifying CICS Trace Selection Parameters

File Options View Services Help

---

Format CICS Auxiliary Data Set Line 1 Col 1 80  
Command ==> Scroll ==> CSR

Enter the name of a CICS auxiliary trace data set and press Enter to initiate formatting. To return from this display without formatting trace, issue the Exit (PF3) or Cancel (PF12) command.

Trace Data Set Name . . . . : 'CICS.TRACE1'

\*\*\* Bottom of data.

Processing auxiliary trace data set. Please wait...

*Figure 104. 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 115, 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 related to Java. The Java execution might be under WebSphere, CICS or Unix System Services on MVS. Typically, the environment will be 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.

---

### Setting options for Java analysis

The general interactive reanalysis options are also used for Java analysis—see “Interactive reanalysis options” on page 97.

---

### Selecting a Java dump data set

Specification of the 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 58). This 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 163.

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 105 on page 176.

## Selecting a Java dump data set

Create Java Fault Entry

Line 1 Col 1 76

Command ==> \_\_\_\_\_

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 105. 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 will be created and the analysis will continue with DTFJ processing performed asynchronously. However, Java information will not be complete until the DTFJ processing has ended, thus requiring a subsequent reanalysis of the fault entry to be performed. When exiting the interactive reanalysis report following the initial analysis, the specified history file will be 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 will still continue. In this case, the dump analysis will be performed as if no Java activity was detected, and the user will be prompted to create a fault entry when exiting the analysis instead.

---

## Java fault entry reanalysis

Only fault entries which were created from analysis of Java dumps, or from real-time analysis of Java abends, will 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 22 and “Invoking Fault Analyzer from Java dump events” on page 24.

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 106 on page 177.

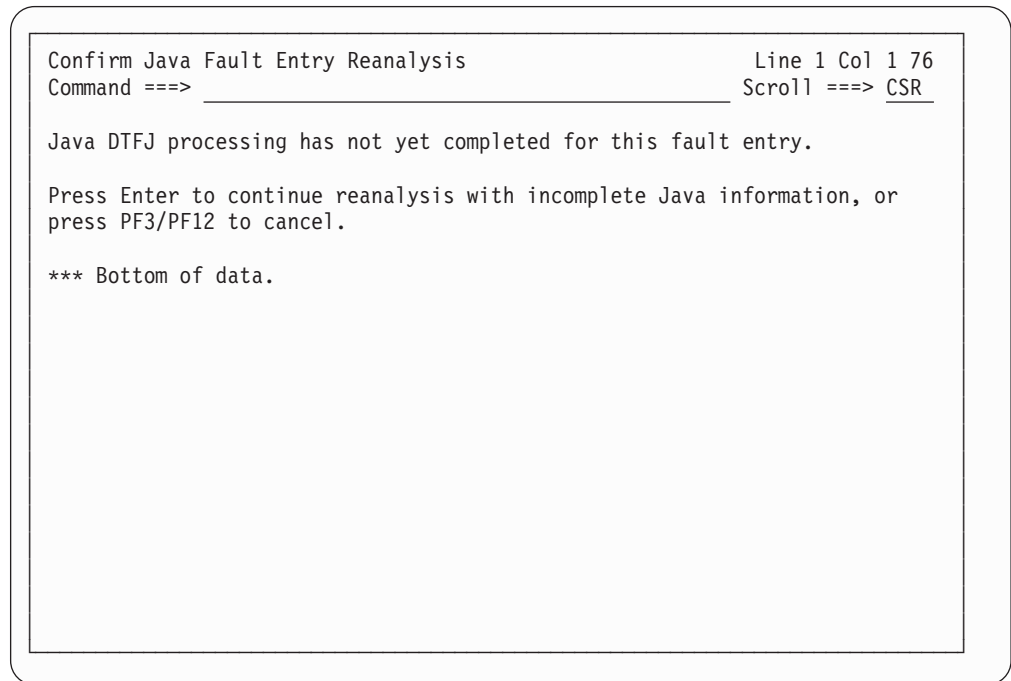


Figure 106. Sample Confirm Java Fault Entry Reanalysis display

By pressing the Enter key, reanalysis will continue, but the Java information will be incomplete.

If instead PF3 or PF12 is pressed, then the reanalysis will be canceled. 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.

**Note:** If DTFJ processing never completes, then the reason might be that 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 237.

## 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 107 on page 178.

## Displaying the Java information in the interactive report

```
File View Services Help
Interactive Reanalysis Report
Command ==>
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 107. 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 108 on page 179.



## Displaying the Java information in the interactive report

File View Services Help							
Event Summary					Top of data		
Command ==>					Scroll ==> CSR		
JOBNAME: BPXBATC3    SYSTEM ABEND: 0CB    FAE3    2010/03/16 19:25:50							
{The following events are presented in chronological order.}							
Event #	Type	Fail Point	Module Name	Program Name	EP Name	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		CEEBINIT	n/a	CEEOPCMM	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		libMyD11Lib.so		*Java_Ja Java_Java2C2CobolExample_callCobol	P+3F8 E+90	/u/rtu
13	Call		libMyD11Lib.so		*Java_Ja doSomething1	P+340 E+90	/u/rtu
14	Call		libMyD11Lib.so		*Java_Ja doSomething2	P+288 E+90	/u/rtu
15	Call		libMyD11Lib.so		*Java_Ja doSomething3	P+142 E+92	/u/rtu
16	Abend S0CB	*****	libMyD11Lib.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 108. 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 109.

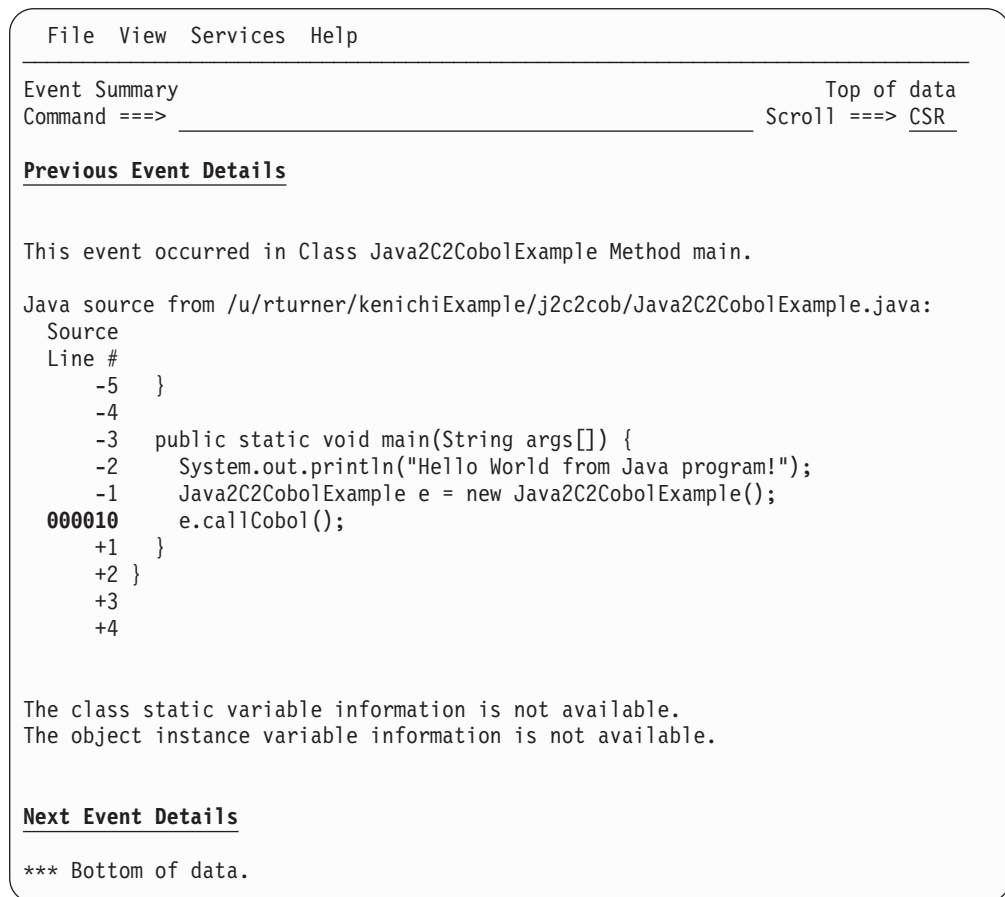


Figure 109. Sample Java Event Details display

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 110 on page 181.

```

File View Services Help

Java Information
Command ==>
JOBNAME: BPXBATC2  SYSTEM ABEND: 0CB  FAE2  2009/10/02  06:53:54
Line 1 Col 1 80
Scroll ==> CSR

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 110. 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 110. Sample Java Information display (Part 2 of 2)

---

## Chapter 9. The Fault Analyzer report

This chapter describes the contents of the report 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 related to CICS system abends, WebSphere, or Java analysis, that is not included in the real-time or batch analysis reports. For further details, see Chapter 6, “Performing CICS systemabend dump analysis,” on page 163.

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 run-time 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 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 will contain “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 111 on page 184 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 111. Sample real-time report extract of open input file information

Figure 112 on page 185 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 112. 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.

### 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 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, WebSphere, or Java environment information, then a note is added here to inform the user that the better way to analyze the current fault is by using the appropriate ISPF interface reanalysis method. This is 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 provided in the details section for the event.

The following is provided for each entry in the list:

#### Event #

This 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.



**Execute**

An EXECUTE event.

**IDISNAP**

An IDISNAP event.

**Interrupt**

An interrupt event.

**Java**

A Java event.

**LE Condition**

An Language Environment condition event.

**Link**

A LINK event.

**ONCODE** *code*

A PL/I ONCODE event.

**Prog Int**

A program interruption event.

**SVC** *number*

An SVC event.

**Fail Point**

If the event has been identified as the point of failure, then this is indicated by \*\*\*\*\* in this column.

**Module Name**

If available, the name of the load module associated with the event.  
Otherwise, n/a.

**Program Name**

If available, the name of the program or CSECT associated with the event.  
Otherwise, n/a.

**EP Name**

If available, the name of the entry point associated with the event.  
Otherwise, n/a.

**Event Location**

One or more of the following abbreviations might appear in this field:

**F#n** Source file number

**L#n** Source file line number

**S#n** Listing file statement number

**M+x** Offset from start of load module

**P+x** Offset from start of program

**E+x** Offset from start of entry point

**Loaded From**

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.

If the data set name cannot be determined for a module not in LPA or CSA, then Not determined will be shown.

**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

## Main report sections

events will be 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 additional information 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 will also find information such as abend code explanations and open file buffers here.

Source code information shown in the details section of the report will include 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 additional source statements are in addition to the statements caused by the macro expansion.

To change the default additional 5 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 377).

If no matching compiler listing or side file was provided for the point-of-failure event, then the failing machine instruction will be shown. In addition, up to 12 instructions ahead of, and 6 following, the failing instruction will be provided. An example of this follows, which is based on the IDIVPCOB COBOL IVP, but with no compiler listing or side file provided:

```
<H2> EVENT 1 OF 1: ABEND S0C7
```

```
*****
***** P O I N T   O F   F A I L U R E *****
*****
```

```
Abend Code. . . . . : S0C7
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 IDISCLB1.

```
Load Module Name. . . . . : SYS05291.T124505.RA000.IDIVPCOB.G0SET.H02(IDISCLB1)
  At Address. . . . . : 16700D88
  Load Module Length. . . . : X'1278'
  Link-Edit Date and Time . : 2005/10/18 12:45:07
```

```
Program and Entry Point Name: IDISCLB1
  At Address. . . . . : 16700D88 (Module IDISCLB1 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 IDISCLB1 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 IDISCBL1 program IDISCBL1 + X'209')
R2: 0001B7FC (231428 bytes of storage addressable)
R3: 16701126 (Module IDISCBL1 program IDISCBL1 + X'39E')
R4: 16700DC0 (Module IDISCBL1 program IDISCBL1 + 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 IDISCBL1 program IDISCBL1 WORKING-STORAGE SECTION
    BLW=00000 + X'0')
R9: 1678C100 (253696 bytes of storage addressable)
R10: 16700E9C (Module IDISCBL1 program IDISCBL1 + X'114')
R11: 16700FBC (Module IDISCBL1 program IDISCBL1 + X'234')
R12: 16700E84 (Module IDISCBL1 program IDISCBL1 + X'FC')
R13: 0002A018 (172008 bytes of storage addressable)
R14: 967010D2 (Module IDISCBL1 program IDISCBL1 + 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.

## The abend job information section

This section provides information about the abending job 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
```

## Main report sections

### The options section

This 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 allocated during the analysis, followed by the fault ID assigned. A batch reanalysis report will instead provide information about the fault ID and history file 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 IDL.SIDIDOC1 data set as shown in the following.

*Table 5. Sample reports*

Description	IDL.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 IDL.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 IDL.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 IDL.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 IDL.SIDISAM1.	IDISRP04
MQSeries. This report was produced by performing reanalysis of a fault entry created for an MQSeriesabend.	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 IDL.SIDISAM1.	IDISRP06

## Sample reports

---

## 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 445 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 installed, then a Fault Analyzer client is available 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 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 additional information about using this interface.

---

### Performing interactive reanalysis under CICS

Fault Analyzer uses an additional 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 31), 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 will be detected on the next invocation of the main CICS transaction.

Refer to “Enabling interactive reanalysis under CICS” on page 447 for information about installing this interface.

---

### Using the Fault Analyzer web browser interface

An optional web interface feature is provided with Fault Analyzer, which allows the real-time report of a fault entry to be viewed on a web browser. This feature is limited to viewing the real-time report of a fault entry, and hence if interactive reanalysis is required, then this should be done using the Fault Analyzer ISPF interface.

Refer to “Installing the Fault Analyzer web browser interface” on page 448 for information about installation of this interface.

The web interface can be used in one of three ways:

1. View all the fault entries in a specific fault history data set.
2. View the real-time report of a specific fault entry.
3. View the last ten accessed history files for a given TSO/ISPF user.

**Note:** This method requires that the web interface user knows the ISPPROF data set name where the IDIPROF<sup>6</sup> member is stored. For example, *userid.ISPF.ISPPROF*.

---

6. This assumes that the application IDI used for Fault Analyzer is ID. If a different application ID is used, then the name of the profile member is *appl-idPROF*.



These methods are described in the following and illustrated with appropriate screen shots.

## Viewing all fault entries in a specified history file

To do this, you need to specify the required fault history data set on the URL. This method uses the FAULTANAL service name. For example:

`http://server-name/FAULTANAL/swilken.hist`

This would display the page shown in Figure 113.

All Fault entries for history file 'SWILKEN.HIST'				
<u>Column Configuration</u>				
FAULT_ID	JOB/TRAN	USER_ID	SYS/JOB	ABEND
<a href="#">SW18865</a>	IMSWRGN6	IMSWRGN	SYSD	S0CB
<a href="#">SW18864</a>	IMSWRGN6	IMSWRGN	SYSD	S06F
<a href="#">SW18861</a>	O6SD	O6SD	SY1D	S06F
<a href="#">SW18859</a>	SWILKEN	SWILKEN	FAE1	S0C4
<a href="#">SW18857</a>	SWILKEN	SWILKEN	FAE1	S06F
<a href="#">SW18852</a>	PMR01555	JERRYBL	FAE1	S06F
<a href="#">SW18866</a>	HANKO	HANKO	ISA1	S06F
<a href="#">SW18855</a>	S0DPDCT9	S0DPDCT9	DEV1	SD23
<a href="#">SW18856</a>	S0DPDCT9	S0DPDCT9	DEV1	SD23
<a href="#">SW18847</a>	IDIVPCOB	SWILKEN	FAE2	S0C7
<a href="#">SW18846</a>	IDIVPCOB	SWILKEN	FAE3	S0C7
<a href="#">SW18849</a>	S0DPDCT9	S0DPDCT9	DEV1	S06F
<a href="#">SW18845</a>	IDIVPCOB	SWILKEN	FAE3	S0C7
<a href="#">SW18844</a>	HANKO	HANKO	ISA1	S0C4
<a href="#">SW18853</a>	PMR60296	RTURNER	FAE1	S06F

Figure 113. Sample history file web display

From this page, an individual real-time report can be viewed by clicking on the appropriate fault ID, for example, fault ID F00001 (see “Viewing the real-time report of a specific fault entry” on page 196).

## Changing Headings

Follow the Column Configuration link to change the headings of the history file web display. By default, Fault ID, Job/Tran, User ID, Sys/Job, and Abend are displayed. By changing the headings, several other headings can be added to the display. This method uses the FAULTHDSEL service name. For example:

`http://server-name/FAULTHDSEL/swilken.hist`

This would display the available heading options shown in Figure 114 on page 196.

Choose headers

<input checked="" type="checkbox"/> Fault_ID	<input checked="" type="checkbox"/> Job/Tran	<input checked="" type="checkbox"/> User_ID
<input checked="" type="checkbox"/> Sys/Job	<input checked="" type="checkbox"/> Abend	<input type="checkbox"/> Appl_ID
<input type="checkbox"/> Class	<input type="checkbox"/> CICS_Trn	<input type="checkbox"/> Dups
<input type="checkbox"/> Dup_Date	<input type="checkbox"/> Dup_Time	<input type="checkbox"/> EXEC_Pgm
<input type="checkbox"/> Date	<input type="checkbox"/> History_File_DSN	<input type="checkbox"/> I_Abend
<input type="checkbox"/> IMS_Pgm	<input type="checkbox"/> Job_ID	<input type="checkbox"/> Job_Type
<input type="checkbox"/> Jobname	<input type="checkbox"/> Lock	<input type="checkbox"/> Minidump
<input type="checkbox"/> Module	<input type="checkbox"/> MD_Pages	<input type="checkbox"/> MVS_Dump
<input type="checkbox"/> MVS_Dump_DSN	<input type="checkbox"/> Netname	<input type="checkbox"/> Offset
<input type="checkbox"/> Program	<input type="checkbox"/> Stepname	<input type="checkbox"/> System
<input type="checkbox"/> Task	<input type="checkbox"/> Term_ID	<input type="checkbox"/> Time
<input type="checkbox"/> User_Title	<input type="checkbox"/> Username	

View History File

Reset

Figure 114. Sample column heading selection web display

## Viewing the real-time report of a specific fault entry

A specific real-time report can be viewed by specifying the history file and fault ID in the URL. This method uses the FAULTHIST service name. For example:

`http://server-name/FAULTHIST/swilken.hist(ANS01860)`

This would display the same page as shown in Figure 115 on page 197.

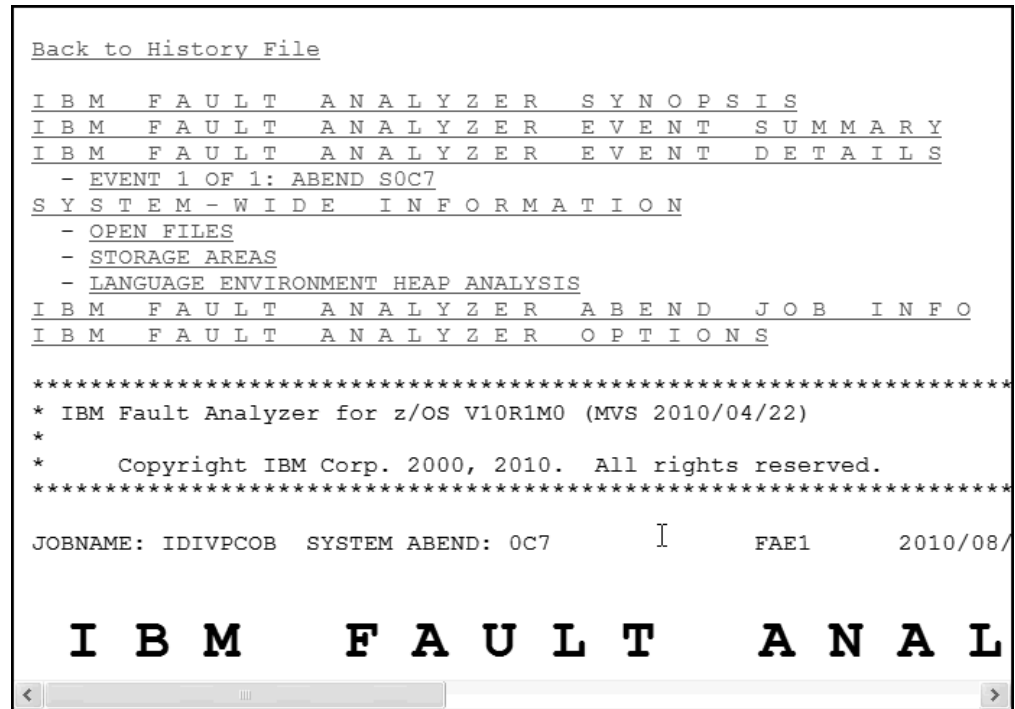


Figure 115. Sample real-time report web display

Specific areas of the report can be easily positioned to by selecting one of the links shown at the top of the page.

## Viewing the last ten accessed history files for a given TSO/ISPF user

By specifying the ISPPROF data set name which contains the IDIPROF member, the last ten history files accessed by the TSO/ISPF user can be displayed. This method uses the FAULTANAL service name. For example:

`http://server-name/FAULTANAL/SWILKEN.FAE1.ISPF.ISPPROF`

This would display the page shown in Figure 116 on page 198.



*Figure 116. Sample last used history files web display*

From this page, a specific history file can be selected for display of its fault entries.

If this method is used to select a history file, then the column configuration extracted from the IDIPROF member will be used, rather than the default column configuration, which is used if the history file data set name is specified directly as in "Viewing all fault entries in a specified history file" on page 195.

---

## Part 2. Fault Analyzer installation and administration

### Chapter 11. Migrating from an earlier version of Fault Analyzer

Migrating from V11.1 to V12.1 . . . . .	205
Migrating from V10.1 to V11.1 . . . . .	205
Migrating from V9.1 to V10.1 . . . . .	205
Migrating from V8.1 to V9.1 . . . . .	206
Migrating from V7.1 to V8.1 . . . . .	206
Migrating from V6.1 to V7.1 . . . . .	207
LPA module compatibility . . . . .	209
Sharing of history files across a sysplex with mixed levels of Fault Analyzer . . . . .	209

### Chapter 12. Preparing to customize Fault Analyzer

Checklist for installing and customizing Fault Analyzer . . . . .	213
Library names after you finish installing . . . . .	216
Storage recommendations . . . . .	218
Exits for invoking Fault Analyzer. . . . .	219
Invocation for non-CICS transaction abends . . . . .	219
Summary of exit usage . . . . .	221
Invocation for CICS transaction abends. . . . .	221
SVC dump registration . . . . .	222
Language Environment options required for invocation of Fault Analyzer . . . . .	222
LE options required for non-CICS abends . . . . .	222
LE options required to capture Java application abends . . . . .	223
LE options required for CICS abends . . . . .	223
Running Fault Analyzer with similar third-party products . . . . .	223
MVS dump data set size. . . . .	224
Application-handled error conditions . . . . .	224

### Chapter 13. Customizing the operating environment for Fault Analyzer

Making Fault Analyzer modules available. . . . .	225
Defining program control access to Fault Analyzer programs. . . . .	226
Restricting change of history file settings . . . . .	226
Setting up the message and abend code explanation repository . . . . .	227
Managing recovery fault recording data set access . . . . .	228
SDUMP recovery fault recording data sets. . . . .	229
Using the XFACILIT resource class for SDUMP RFR data sets . . . . .	229
TDUMP recovery fault recording data sets . . . . .	229
Using the XFACILIT resource class for TDUMP RFR data sets . . . . .	230
RFR TDUMP XFACILIT example. . . . .	231

### Chapter 14. Using the Fault Analyzer IDIS subsystem

Sysplex-wide subsystem inter-communication . . . . .	234
Caching of history file \$INDEX data . . . . .	234

Starting the IDIS subsystem . . . . .	235
IDIS subsystem storage requirements . . . . .	236
IDIS subsystem requirements for DB2 . . . . .	237
IDIS subsystem requirements for Java . . . . .	237
Stopping the IDIS subsystem . . . . .	238

### Chapter 15. Modifying your ISPF environment

Allocating ISPF data sets . . . . .	239
Making the Fault Analyzer IDISCMDS command table available . . . . .	239
Updating the ISPF selection panel . . . . .	240
Invoking Fault Analyzer using an ISPF 3.4 line command (FA) . . . . .	240
Invoking Fault Analyzer from SDSF (SFA command) . . . . .	241
Invoking the LOOKUP command using cursor selection (LOOKC command) . . . . .	241
Providing ISPF interface defaults for new users . . . . .	242
Providing installation-specific batch reanalysis JCL control statements. . . . .	242

### Chapter 16. Customize Fault Analyzer by using USERMODs

Enabling Fault Analyzer to be invoked . . . . .	243
Installing the MVS change options/suppress dump exit IDIXDCAP (++)IDITABX) . . . . .	243
Enabling the Language Environment abnormal termination exit IDIXCEE . . . . .	243
Working with applications that use a non-Language Environment run time . . . . .	244
Identifying the LE run-time library (++)IDILEDs) . . . . .	244
Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++)IDISPLI/++)IDISPLIA) . . . . .	245
Always invoking Fault Analyzer from PL/I PLIDUMP (++)IDISPDM) . . . . .	245
Eliminating the need for a dump DD statement (++)IDITABD) . . . . .	245
Specifying an alternative parmlib data set for IDICNF00 (++)IDISCNF) . . . . .	246
Changing the default recovery fault recording IEATDUMP data set name (++)IDISRFR) . . . . .	246
Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (++)IDISXCUM). . . . .	247
Obtaining load modules from CA-Panexec . . . . .	247

### Chapter 17. Setting up history files

Determining what size history files to allocate . . . . .	249
Allocating a PDS or PDSE for a history file . . . . .	249
AUTO-managed PDSE history files . . . . .	250
Providing the name of a history file to Fault Analyzer . . . . .	250
Setting up views . . . . .	250
Specifying a default column layout . . . . .	252

Specifying an initial fault entry selection criteria	252
View considerations if not using XFACILIT resource class	253
Managing history files across MVS systems without shared DASD	253
Automated implementation	254
On demand implementation	258
Using the on demand implementation	260
Sharing of history files across a sysplex.	260
Managing history file fault entry access	261
Using the XFACILIT resource class for history file fault entries	261
XFACILIT implementation example 1	263
XFACILIT implementation example 2: Using global access table.	263
XFACILIT implementation example 3: Using ACF2	263
Using the IDIXFXIT user exit	264

## Chapter 18. Setting and changing default options for the site

Parmlib member IDICNFxx	267
Controlling which jobs are analyzed with Exclude processing	269
Fast Exclude options processing	270
Controlling report detail.	270
Limiting the size of minidumps	271
Pointing to listings and REXX exec libraries	271
Suppressing noncritical SYSLOG messages	271
Customizing the Fault Entry List display	271
Specifying the cultural environment.	271

## Chapter 19. Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products

Updating your build process	274
Updating your promotion process	275
Preparing your programs	275
Enterprise COBOL for z/OS Version 4 programs	275
Preparing Enterprise COBOL for z/OS Version 4 programs	276
Sample JCL for compiling Enterprise COBOL for z/OS Version 4 programs	277
Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs	278
Preparing Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs.	278
Sample JCL for compiling Enterprise COBOL for z/OS Version 3 programs	279
COBOL for MVS and VM programs.	280
Preparing COBOL for MVS and VM programs.	280
Sample JCL for compiling COBOL for MVS and VM programs.	281
VS COBOL II programs	282
Preparing VS COBOL II programs	283
Sample JCL for compiling VS COBOL II programs.	283
OS/VS COBOL programs	284
Preparing OS/VS COBOL programs.	284

Sample JCL for compiling OS/VS COBOL programs.	285
Enterprise PL/I Version 3.7 and later programs	285
Preparing Enterprise PL/I Version 3.7 and later programs	286
Sample JCL for compiling Enterprise PL/I for z/OS Version 3.7 or later programs	288
Enterprise PL/I Version 3.5 and Version 3.6 programs.	288
Preparing Enterprise PL/I Version 3.5 and Version 3.6 programs.	290
Sample JCL for compiling Enterprise PL/I Version 3.5 or Version 3.6 programs	291
Enterprise PL/I Version 3.4 and earlier programs.	291
Preparing Enterprise PL/I Version 3.4 and earlier programs	292
Sample JCL for compiling Enterprise PL/I for z/OS Version 3.4 or earlier programs	294
PL/I for MVS and VM and OS PL/I programs	294
Preparing PL/I for MVS and VM and OS PL/I programs	294
z/OS XL C and C++ programs	296
Preparing z/OS XL C and C++ programs	298
Sample JCL for compiling z/OS C++ programs.	300
Assembler programs	300
Preparing Assembler programs	301
Sample JCL for assembling a program	301

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

Creating side files using IDILANGX.	303
IDILANGX parameters	305
Side file compatibility with Debug Tool for z/OS	306
Including an IDILANGX step in your SCLM translator.	306
High Level Assembler SCLM example	306
COBOL SCLM example	307
COBOL Report Writer Precompiler	307
Required compiler options for IDILANGX.	308
TEST option considerations.	309
DEBUG option considerations.	310
Naming compiler listings or side files	310
Naming CSECTs for Fault Analyzer	311
Locating compiler listings or side files	311
IDITRACE information	313
Compiler listings and side file attributes	313
Using the IDIRLOAD DDname for CSECT mapping	314
IDILANGP side file formatting utility	315
ISPF-packed compiler listings	316

## Chapter 21. Customizing the CICS environment

Configuring Language Environment for CICS to invoke Fault Analyzer	320
Defining required programs to CICS	320
Adding the required programs to the startup PLT	320
Adding the required programs to the shutdown PLT	321

Enabling dynamic control of analysis of CICS transaction abends. . . . .	321
Using CFA to FORCEPURGE the currently analyzed task . . . . .	321
SVC dump screening. . . . .	321
Sample definition job. . . . .	322
Controlling CICS transaction abend analysis . . . . .	323
Using CFA from a CICS terminal. . . . .	323
Clearing the NoDup(CICSFAST(...)) recording area . . . . .	324
IDITRACE under CICS . . . . .	324
Using CFA from an MVS console. . . . .	325
Ensuring transaction abend analysis is not suppressed by DUMP(NO). . . . .	326
CICS NoDup(CICSFAST) override assembler exit (IDINDFUE). . . . .	326
Invocation . . . . .	326
Entry specifications . . . . .	327
Return specifications . . . . .	327
Example . . . . .	327
Preventing LE from causing the CICS trace to wrap	328
Specifying CICS options through the IDIOPTS DDname . . . . .	328
Language Environment abend considerations. . . . .	329
Capture of abends running on CICS user key open TCBs (L9 TCBs) . . . . .	329
Installing the MVS post-dump exit IDIXTSEL . . . . .	329
Storage requirements. . . . .	329
Maximizing CICS transaction abend analysis performance. . . . .	329
<b>Chapter 22. Customizing the DB2 environment</b>	331
Binding DB2. . . . .	331
DB2 and Language Environment . . . . .	331
DB2 stored procedures . . . . .	331
Improving Fault Analyzer DB2 performance . . . . .	333
<b>Chapter 23. Customizing the IMS environment</b>	335
IMS and Language Environment . . . . .	335
<b>Chapter 24. Customizing the Fault Analyzer Japanese feature</b> . . . . .	337
Allocating ISPF data sets . . . . .	337
Setting the national language . . . . .	337
<b>Chapter 25. Customizing the Fault Analyzer Korean feature</b> . . . . .	339
Allocating ISPF data sets . . . . .	339
Setting the national language . . . . .	339
<b>Chapter 26. Verifying the customization of Fault Analyzer</b> . . . . .	341
Verifying the use of Fault Analyzer with assembler	341
Verifying the use of Fault Analyzer with COBOL	342
Verifying the use of Fault Analyzer with PL/I . . . . .	343
Verifying the use of Fault Analyzer with C . . . . .	344
Verifying the IDIXCEE Language Environment exit enablement . . . . .	345
Verifying the IDITABD USERMOD installation . . . . .	345
Verifying the customization of Fault Analyzer under CICS . . . . .	345

CICS IVP: 0C1 in program IDIXFA . . . . .	346
CICS IVP: EXEC CICS DUMP DUMPCODE(FAD1) . . . . .	346
CICS IVP: EXEC CICS ABEND ABCODE(FLT1)	347
CICS IVP: EXEC CICS ABEND ABCODE(FLT2)	347
Verifying the use of Fault Analyzer with DB2 . . . . .	347
Using a C program . . . . .	347
Using a COBOL program . . . . .	349
Verifying the use of Fault Analyzer through ISPF	351
Verifying the recovery fault recording set-up . . . . .	351

<b>Chapter 27. Managing history files (IDIUTIL utility)</b> . . . . .	353
IDIUTIL control statements. . . . .	354
FILES control statement . . . . .	354
Description . . . . .	354
LISTHF control statement . . . . .	354
Description . . . . .	355
DELETE control statement . . . . .	355
Description . . . . .	355
SETFAULTPREFIX control statement . . . . .	356
Description . . . . .	356
SETMAXFAULTENTRIES control statement . . . . .	357
Description . . . . .	357
IMPORT control statement . . . . .	358
Description . . . . .	358
EXITIS control statement . . . . .	359
Description . . . . .	359
Examples. . . . .	360
Example 1. Listing history file entries . . . . .	360
Example 2. Deleting history file entries by date	360
Example 3. Deleting history file entries by utilization . . . . .	360
Example 4. Changing history file fault prefix characters . . . . .	360
Example 5. Creating a self-maintained history file . . . . .	361
Example 6. Importing history file entries . . . . .	361
IDIUTIL batch utility user exit samples. . . . .	362

<b>Chapter 28. Providing explanations for application-specific messages</b> . . . . .	363
Message search order. . . . .	364
Refreshing cached messages . . . . .	364

<b>Chapter 29. Maintaining Fault Analyzer</b> . . . . .	365
---	-----

<b>Chapter 30. Disabling Fault Analyzer.</b> . . . .	367
Temporarily deinstalling Fault Analyzer . . . . .	367
Turning off Fault Analyzer using the IFAPRDxx parmlib member . . . . .	367
Turning off Fault Analyzer with a JCL switch (IDIOFF) . . . . .	368
Turning off Fault Analyzer using an environment variable (_IDI_OFF) . . . . .	368

<b>Chapter 31. Customizing Fault Analyzer by using user exits</b> . . . . .	369
Invocation parameters . . . . .	372
Global environment data area (ENV) . . . . .	372
User exit type specific data area . . . . .	372



Supported exit programming languages . . . . .	372	When invoked . . . . .	412
Data area version checking . . . . .	373	Parameters . . . . .	412
Diagnostic tracing . . . . .	373	Example . . . . .	413
Tracing user exit parameter list values . . . . .	373	IDIUTIL Delete user exit . . . . .	413
Tracing REXX EXECs. . . . .	377	Purpose . . . . .	413
Descriptions of fault analysis user exit types . . . . .	377	When invoked . . . . .	413
Analysis Control user exit . . . . .	377	Parameters . . . . .	413
Purpose . . . . .	377	Example . . . . .	414
When invoked . . . . .	379	IDIUTIL ListHF user exit . . . . .	414
Parameters . . . . .	379	Purpose . . . . .	414
Example . . . . .	380	When invoked . . . . .	415
Analysis Control user exit (Dump registration) . . . . .	380	Parameters . . . . .	415
Purpose . . . . .	381	Example 1 . . . . .	415
When invoked . . . . .	381	Example 2 . . . . .	416
Parameters . . . . .	381	User exit REXX commands . . . . .	418
Example . . . . .	381	Evaluate command . . . . .	418
Compiler Listing Read user exit . . . . .	382	Parameters . . . . .	418
Purpose . . . . .	382	Return codes . . . . .	420
When invoked . . . . .	385	Example . . . . .	420
Parameters . . . . .	385	IDIALLOC command. . . . .	420
Example . . . . .	385	Return codes . . . . .	423
Message and Abend Code Explanation user exit . . . . .	386	Example . . . . .	423
Purpose . . . . .	386	IDIDTEST command . . . . .	424
When invoked . . . . .	389	Return codes . . . . .	424
Parameters . . . . .	389	Example . . . . .	424
Example . . . . .	390	IDIDSECTdsn command. . . . .	424
Formatting user exit . . . . .	390	Return codes . . . . .	425
Purpose . . . . .	390	Example . . . . .	425
When invoked . . . . .	392	IDIEventInfo command . . . . .	425
Parameters . . . . .	392	Parameters . . . . .	425
Example 1 . . . . .	392	Return codes . . . . .	425
Example 2 . . . . .	395	Example . . . . .	426
Example 3 (Hogan) . . . . .	397	IDIFREE command . . . . .	426
Example 4 (Batch MVS dump registration) . . . . .	401	Return codes . . . . .	426
End Processing user exit. . . . .	402	Example . . . . .	426
Purpose . . . . .	402	IDIModQry command . . . . .	426
When invoked . . . . .	403	Return codes . . . . .	427
Parameters . . . . .	403	Example . . . . .	427
Example . . . . .	404	IDIRegisterFaultEntry command . . . . .	427
End Processing user exit (Fault entry refresh) . . . . .	404	Return codes . . . . .	428
Purpose . . . . .	404	Example . . . . .	428
When invoked . . . . .	405	IDIWRITE command . . . . .	428
Parameters . . . . .	405	Return codes . . . . .	429
Example . . . . .	405	Example . . . . .	429
Notification user exit . . . . .	405	IDIWTO command . . . . .	429
Purpose . . . . .	405	Return codes . . . . .	430
When invoked . . . . .	406	Example . . . . .	430
Parameters . . . . .	406	List command . . . . .	430
Example 1 . . . . .	407	Parameters . . . . .	430
Example 2 . . . . .	408	Return codes . . . . .	431
Example 3 . . . . .	409	Example . . . . .	431
Example 4 . . . . .	410	Note command. . . . .	432
Notification user exit (Dump registration) . . . . .	411	Parameters . . . . .	432
Purpose . . . . .	411	Return codes . . . . .	432
When invoked . . . . .	411	Example . . . . .	433
Parameters . . . . .	411	Formatting tags . . . . .	433
Example . . . . .	411	ADDR (address) . . . . .	436
Descriptions of IDIUTIL batch utility user exit types . . . . .	412	Description . . . . .	436
IDIUTIL Import user exit . . . . .	412	AREA (area). . . . .	436
Purpose . . . . .	412	Description . . . . .	437
		DD (definition description). . . . .	437



Description . . . . .	437
DATA (data). . . . .	437
Description . . . . .	437
DL (definition list). . . . .	437
Description . . . . .	438
DT (definition term) . . . . .	438
Description . . . . .	439
DUMP (EBCDIC dump). . . . .	439
Description . . . . .	439
DUMPA (ASCII dump) . . . . .	439
Description . . . . .	440
HP (highlighted phrase). . . . .	440
Description . . . . .	440
L (line) . . . . .	440
Description . . . . .	441
LI (list item). . . . .	441
Description . . . . .	441
NOTEL (note list) . . . . .	441
Description . . . . .	441
P (paragraph) . . . . .	441
Description . . . . .	442
TH (table heading) . . . . .	442
Description . . . . .	442
U (underline) . . . . .	442
Description . . . . .	443
UL (unordered list) . . . . .	443
Description . . . . .	443
 <b>Chapter 32. Installing non-ISPF interfaces to access Fault Analyzer history files . . . . .</b>	 445
Installing the Fault Analyzer plug-in for Eclipse . . . . .	445
Customize the IBM Problem Determination Tools for z/OS Common Component Common Server . . . . .	445
Download and install CICS Explorer . . . . .	446
Download Fault Analyzer plug-in . . . . .	446
Install Fault Analyzer plug-in into CICS Explorer . . . . .	446
Installing the Fault Analyzer client for IBM Rational Developer for System z . . . . .	446
Enabling interactive reanalysis under CICS . . . . .	447
Making the required CICS resource definitions . . . . .	447
Making the required CICS JCL changes. . . . .	447
Installing the Fault Analyzer web browser interface . . . . .	448
Running the sample job to create a program object in a z/OS Unix System Services file. . . . .	448
Making the required updates to the HTTP server configuration files . . . . .	448



---

## 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 209 and “Sharing of history files across a sysplex with mixed levels of Fault Analyzer” on page 209 is applicable to all versions of Fault Analyzer.

---

### 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.

- 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 IDLSIDIAUT2 must be added as STEPLIB in the IDIS subsystem start-up JCL. For details, see “Starting the IDIS subsystem” on page 235.
- Users of Message and Abend Code Explanation load module user exits should note that the offset to the XPL.ABEND\_CODE field has changed.
- Prompting for missing compiler listings or side files during interactive reanalysis previously 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. This could have meant that prompting occurred for some fault entries, but not for others.

With the user-specified option now available to control prompting (see “Interactive reanalysis options” on page 97), 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 either specifying one or more compiler listing or side file data sets in the IDICNFxx parmlib member (in which case the default is to prompt), or by ensuring 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 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 58. 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 been increased.
- The WZClient option has been renamed to RDZClient. However, the WZClient 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 for the purpose of providing message and abend code explanations. Instead, a VSAM cluster is populated with this information.

Changes related to this include:

- Allocating and populating the new VSAM cluster. For details, see “Setting up the message and abend code explanation repository” on page 227.
- Deletion of the old cache data set to reclaim DASD space.
- 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.

Changes related to this 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, additional 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 will 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 377).
- 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 228).
- The NoDup(ImageFast(0)) default option has changed to NoDup(ImageFast(5)). If IMS fast duplicate detection is not desired, then it will be 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 this is not desired, then new options have been provided to override the defaults. For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 233.
- If a version of Fault Analyzer prior to V8.1 will be used in parallel with V8.1, then either AUTO-managed history files should not be used (default for new PDSE history files with V8.1, or set using the IDIUTIL SetMaxFaultEntries(AUTO) control statement), or the applicable compatibility PTF be installed 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 will be attempted invoked.

Review all specifications of Exits or DumpRegistrationExits options to ensure that correct processing will occur 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 492).
  - 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 481), or control the writing of the minidump using an End Processing user exit instead.

## Migrating from V6.1 to V7.1

- EPC.SUPPRESS\_SYSDUMP  
Use EPC.SUPPRESS\_DUMP instead.
- EPC.DUPLICATE\_FAULT\_ID  
Use ENV.FAULT\_ID instead.
- EPC.SUPPRESS\_IDIMSG  
Use the Quiet option instead (for details, see “Quiet” on page 492).
- NFY.SUPPRESS\_IDIMSG  
Use the Quiet option instead (for details, see “Quiet” on page 492).

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 will be 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 will be 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. This will generally result in the inability 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 will be necessary to override the default by, for example, adding the following option to your IDICNF00 parmlib member, of an IDIOPTS user options file 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.  
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 321.

## LPA module compatibility

Fault Analyzer provides downward compatibility between load modules in the IDLSIDIALPA 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.

## Sharing of history files across a sysplex with mixed levels of Fault Analyzer

If history files are shared between images of a sysplex with different levels of Fault Analyzer installed, then there might be compatibility issues which should be understood, and where possible, eliminated by installing maintenance on the older levels.

To determine the compatibility issues that might be applicable to an installation, use Table 6 as follows:

1. Locate the row matching the most current or proposed version and maintenance level of Fault Analyzer on any image of the sysplex using the "Version" and "Level" columns.

**Note:** Both columns are in reverse chronological order—all APARs and PTFs shown either pre-reqs or supersedes maintenance shown later/below for the same version.

2. Locate the row matching an older level of Fault Analyzer on a different image of the sysplex.
3. The issues that are applicable between the two levels are those described in the column "Compatibility issue with older levels" in all intermediate rows—including the row located in 1, but excluding the row located in 2.

Where maintenance has been recommended to be installed on the older level of Fault Analyzer, the most current APAR/PTF of the applicable set can be determined using the "Version" and "Level" columns—the top-most APAR/PTF of the set is the one which should be installed.

The above steps should be repeated for each image of the sysplex which is not at the most current level.

*Table 6. Fault Analyzer history file compatibility*

Version	Level	Compatibility issue with older levels
V12.1	GA	None.
V11.1	PM54422/UK76660 PM47919/UK74578 PM44693/UK71656 PM42267/UK70024 PM34295/UK68814 PM28208/UK65276 PM27281/UK62719 PM26282/UK62204 GA	None.

## Sharing of history files across a sysplex with mixed levels of Fault Analyzer

Table 6. Fault Analyzer history file compatibility (continued)

Version	Level	Compatibility issue with older levels
V10.1	PM54423/UK76704	None.
	PM47920/UK74641	
	PM44694/UK71798	
	PM42267/UK70115	
	PM34296/UK68854	
	PM27330/UK65177	
	PM21096/UK62342	
	PM14420/UK59774	
	PM11905/UK56713	
	PM08932/UK55782	
	PM03974/UK54542	
	PM00092/UK52832	
	GA	
V9.1	PM54424/UK77122	None.
	PM47921/UK74672	
	PM44695/UK71840	
	PM42269/UK70170	
	PM34297/UK68906	
	PM27336/UK65326	
	PM21127/UK62246	
	PM14421/UK59799	
	PM11905/UK56765	
	PM08936/UK55854	
	PM03975/UK54591	
	PK97412/UK52917	
	PK92822/UK50286	
	PK89333/UK48699	
	PK83693/UK47365	
	PK79442/UK45132	
	PK74751/UK43309	
	PK74114/UK40989	
	PK72608/UK40540	
V9.1	GA	<p><b>Issue:</b> Fault Analyzer abends during reanalysis of fault entries containing user notes.</p> <p><b>Solution:</b> To prevent Fault Analyzer abends, the following PTFs should be installed:  V8.1: UK40617  V7.1: UK40623  V6.1: UK40646</p> <p>At the above or later maintenance levels, abends due to incompatible user note formats should no longer occur. However, user notes created at the Fault Analyzer V9.1 GA level, or later, will be disabled for use with older levels.</p>



## Sharing of history files across a sysplex with mixed levels of Fault Analyzer

Table 6. Fault Analyzer history file compatibility (continued)

Version	Level	Compatibility issue with older levels
V8.1	PM43314/UK70024 PM34298/UK68956 PM27339/UK65329 PM21195/UK62372 PM14434/UK59802 PM11913/UK56862 PM08940/UK55892 PM03979/UK54631 PK97413/UK52922 PK92823/UK50298 PK89334/UK48746 PK83694/UK47455 PK79443/UK45219 PK74745/UK43372 PK74115/UK41024 PK72175/UK40617 PK69687/UK39318 PK67040/UK38412 PK64683/UK36737 PK64242/UK35659 PK61873/UK35125	None.
V8.1	PK56115/UK33904	<p><b>Issue:</b> Fault entries might be unlocked and subsequently inadvertently deleted.</p> <p>If a fault entry has been locked at this or a later level, then it will not be recognized as locked by older levels.</p> <p>A lock value set by an older level will only be recognized by this or a later level, if the value was set before the history file was first updated at this or a later level.</p> <p><b>Solution:</b> To provide lock flag compatibility with older levels of Fault Analyzer, the following maintenance should be installed:  V8.1: UK33904  V7.1: PK74739  V6.1: PK74736</p>
V8.1	PK53632/UK30777	None.
V8.1	GA	<p><b>Issue:</b> Risk of clearing out history files.</p> <p><b>Solution:</b> Either AUTO-managed history files should not be used (default for new PDSE history files as of V8.1 GA, or set using the IDIUTIL SetMaxFaultEntries(AUTO) control statement), or the following PTFs should be installed:  V8.1: UK30777  V7.1: UK30778  V6.1: UK30911</p>
V7.1	Any	None.
V6.1	Any	None.



---

## 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 13 and Chapter 3, “The Fault Analyzer ISPF interface,” on page 31 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 303 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 225.

— 2. **Allocate a history file**

Although multiple history files might eventually be used at your site, a single history file is sufficient for the purpose of verifying 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

## Checklist for installing and customizing Fault Analyzer

through the DataSets option. You will be reviewing the IDICNF00 parmlib member, and the options it might contain, later in the installation process. A suggested size of the initial history file is 100 cylinders.

General information about history files is provided in Chapter 17, “Setting up history files,” on page 249, 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 18, “Setting and changing default options for the site,” on page 267.

Ensure that a

`DataSets(IDIHIST(dsn))`

option is included if you allocated a history file in step 2 on page 213 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 227.

### — 5. Install the MVS change options/suppress dump exit IDIXDCAP with USERMOD IDITABX

For details, see “Installing the MVS change options/suppress dump exit IDIXDCAP (++)IDITABX)” on page 243.

Information about the characteristics of this exit are provided in “Exits for invoking Fault Analyzer” on page 219.

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 243.

For information to help you determine the applicability of this exit at your site, see “Exits for invoking Fault Analyzer” on page 219 and “Language Environment options required for invocation of Fault Analyzer” on page 222.

If this exit is not installed, then abends in LE-enabled programs will only be 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 USERMOD IDITABD to eliminate the need for jobs to include an MVS dump DD statement

For details, see “Eliminating the need for a dump DD statement (++)IDITABD)” on page 245.

### — 8. 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 319.

### — 9. Customize the DB2 environment

## Checklist for installing and customizing Fault Analyzer

This step is only applicable if you are using DB2.

For details, see Chapter 22, “Customizing the DB2 environment,” on page 331.

**Note:** It is recommended that the DB2 table index discussed in “Improving Fault Analyzer DB2 performance” on page 333 is created, as severe Fault Analyzer performance degradation might otherwise result when accessing DB2 catalog information.

### \_\_\_ 10. **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 335.

### \_\_\_ 11. **Customize for ISPF**

For details, see Chapter 15, “Modifying your ISPF environment,” on page 239.

### \_\_\_ 12. **Start the Fault Analyzer IDIS subsystem**

For details, see Chapter 14, “Using the Fault Analyzer IDIS subsystem,” on page 233.

Optional installation steps

### \_\_\_ 13. **Add BPX security server program control profile for Fault Analyzer programs**

This 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 226.

### \_\_\_ 14. **Install USERMOD IDILEDS to identify the name of the LE run-time library (optional)**

For details, see “Identifying the LE run-time library (++IDILEDS)” on page 244.

### \_\_\_ 15. **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 245.

### \_\_\_ 16. **Install USERMOD IDISRFR to change the default recovery fault recording IEATDUMP data set name (optional)**

For details, see “Changing the default recovery fault recording IEATDUMP data set name (++IDISRFR)” on page 246.

### \_\_\_ 17. **Define XFACILIT resource classes to manage recovery fault recording data sets (optional)**

For details, see “Managing recovery fault recording data set access” on page 228.

### \_\_\_ 18. **Customize for Japanese language support**

This is only required if the Japanese feature of Fault Analyzer has been installed.

For details, see Chapter 24, “Customizing the Fault Analyzer Japanese feature,” on page 337.

### \_\_\_ 19. **Customize for Korean language support**

This is only required if the Korean feature of Fault Analyzer has been installed.

## Checklist for installing and customizing Fault Analyzer

For details, see Chapter 25, “Customizing the Fault Analyzer Korean feature,” on page 339.

\_\_\_ 20. **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 445.

\_\_\_ 21. **Grant history file administrator authorization for change of settings via ISPF interface**

For details, see “Restricting change of history file settings” on page 226.

### Installation verification

\_\_\_ 22. **Perform assembler IVP**

For details, see “Verifying the use of Fault Analyzer with assembler” on page 341.

\_\_\_ 23. **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 342.

\_\_\_ 24. **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 343.

\_\_\_ 25. **Perform IDIXCEE Language Environment exit IVP**

For details, see “Verifying the IDIXCEE Language Environment exit enablement” on page 345.

\_\_\_ 26. **Perform IDITABD USERMOD IVP**

For details, see “Verifying the IDITABD USERMOD installation” on page 345.

\_\_\_ 27. **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 345.

\_\_\_ 28. **Perform DB2 IVP**

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 347.

\_\_\_ 29. **Perform ISPF IVP**

For details, see “Verifying the use of Fault Analyzer through ISPF” on page 351.

Additional customization can optionally be performed using user exits as described in Chapter 31, “Customizing Fault Analyzer by using user exits,” on page 369. 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
---------------	------------

## Library names after you finish installing

<b>IDI.SIDIALPA</b>	Load modules that must be made available from LPA.
<b>IDI.SIDIAUTH</b>	Authorized load modules that must be made available from LINKLIST.
<b>IDI.SIDIDOC1</b>	Message and abend code explanation override files, sample reports, and Fault Analyzer User's Guide and Reference PDF.
<b>IDI.SIDIDOC2</b>	Message and abend code explanation repository input data.  The only purpose of this data set is to be used as input by the IDISVENU job described on page 227, which allocates and initializes the message and abend code repository.
<b>IDI.SIDIEXEC</b>	REXX execs.
<b>IDI.SIDILPA1</b>	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.
<b>IDI.SIDIMAPS</b>	Control block maps.
<b>IDI.SIDIMLIB</b>	ISPF message members.
<b>IDI.SIDIMOD1</b>	Non-authorized load modules that must be made available from LINKLIST.
<b>IDI.SIDIPLIB</b>	ISPF panels.
<b>IDI.SIDISAM1</b>	Softcopy samples and installation jobs.
<b>IDI.SIDISLIB</b>	ISPF skeletons.
<b>IDI.SIDITLIB</b>	ISPF tables.

### Japanese feature note

If the Japanese feature of Fault Analyzer has been installed, the following additional data sets should exist:

<b>Data Set Name</b>	<b>Containing</b>
<b>IDI.SIDIDJPN</b>	Japanese message and abend code explanation override files, sample reports, and Fault Analyzer User's Guide and Reference PDF.
<b>IDI.SIDIMJPN</b>	Japanese ISPF message members.
<b>IDI.SIDIPJPN</b>	Japanese ISPF panels.
<b>IDI.SIDISJPN</b>	Japanese ISPF skeletons.
<b>IDI.SIDITJPN</b>	Japanese ISPF tables.
<b>IDI.SIDIXJPN</b>	Japanese softcopy samples and installation jobs.

### End of Japanese feature note

## Library names after you finish installing

### Korean feature note

If the Korean feature of Fault Analyzer has been installed, the following additional 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 occurs, additional 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 SYSMDUMP 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 8 megabytes, and Fault Analyzer in LPA around 11 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 will be issued and processing terminates.

If the necessary above-the-line (31-bit) size is not available, then message IDI0055E will be 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 11 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 to the LPALSTxx parmlib member (for details, see “Making Fault Analyzer modules available” on page 225).

**Note:** If Fault Analyzer modules are loaded into LPA, then it is important that step 2 on page 365 is performed after apply of any Fault Analyzer maintenance.

Failure to perform this step following the installation of maintenance will prevent the update of Fault Analyzer LPA modules. Because all Fault Analyzer modules are not in LPA, this can cause a mismatch between the old and the new code, which might lead to undefined behavior.

The MVS IEFUSI exit may be used as a general way to provide additional 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).

**MVS change options/suppress dump IEAVTABX CSECT exit IDIXDCAP**

Characteristics:

## Exits for invoking Fault Analyzer

- This exit can be used with both Language Environment and non-Language Environment based batch application programs.
- 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>7</sup> will prevent the MVS IEAVTABX exit from being called (for example, a CANCEL command resulting in an Sx22 abend for which most MVS systems will 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 captured using this exit can be performed if a minidump was written, or if a SYSMDUMP DDname was allocated by the abending job step, and the SYSMDUMP data set is available.
- LE-enabled abends will need to run with TERMTHDACT, specifying the suboption UATRACE, UADUMP, UAONLY, or UAIMM, in the LE options so that LE will call a system dump to activate this exit. All other TERMTHDACT suboption settings will 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 (==IDITABX)” on page 243.

### 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 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 will be invoked instead of the LE abnormal termination exit.
- Permits instance-specific LE message inserts to be obtained, and thus included in the analysis report.

For information about installation of this exit, refer to “Enabling the Language Environment abnormal termination exit IDIXCEE” on page 243.

**Note:** This exit is easily confused with LE's CICS abnormal termination exit, CEEEXTAN (Fault Analyzer exit IDIXCEE). Pay special attention if both this and the CICS exits are installed.

---

7. 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.

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 will intercept 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

The following tables indicate the exit used to invoke Fault Analyzer, depending on execution environment, options in effect, specification of MVS dump DD statement, and the IDITABD USERMOD.

*Table 7. Non-Language Environment programs*

IDITABD USERMOD installed	SYSABEND, SYSUDUMP, or SYSMDUMP DD provided	
	Yes	No
Yes	IDIXDCAP	IDIXDCAP
No	IDIXDCAP	Fault Analyzer not invoked

*Table 8. Language Environment programs with TERMTHDACT(UA\*) option in effect*

IDITABD USERMOD installed	SYSABEND, SYSUDUMP, or SYSMDUMP DD provided	
	Yes	No
Yes	IDIXDCAP	IDIXDCAP
No	IDIXDCAP	See Table 9

*Table 9. Language Environment programs without TERMTHDACT(UA\*) option in effect*

IDIXCEE exit installed	Exit used
Yes	IDIXCEE
No	Fault Analyzer not invoked

## 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 are 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 CEEEXTAN CSECT exit, IDIXCEE, described below.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults captured using this exit can be performed if a minidump was written.

## Exits for invoking Fault Analyzer

For information about installation of this exit, refer to Chapter 21, “Customizing the CICS environment,” on page 319.

### **CICS LE abnormal termination CEEEXTAN CSECT exit IDIXCCEE**

Characteristics:

- This exit is only effective with Language Environment based CICS application programs that abend with LE U1xxx or U4xxx-type abends (for example, message IGZ0006S abend U4036 as the result of exceeding the array bounds in a COBOL application compiled using the SSRANGE option).
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults 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 320.

## **SVC dump registration**

One additional 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 233.
- 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 329.

---

## **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 IEAVTABX MVS 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 will permit the IEAVTABX exit to gain control when the MVS dump is about to be written.

## Language Environment options required for invocation of Fault Analyzer

**Note:** If both the CEEEXTAN and IEAVTABX exits are installed, then Fault Analyzer will resolve 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 328.

---

## 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) by ensuring 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 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.

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 will 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

Application error handlers may be written 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 wish 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 18).

An example of an application error handling routine that will 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 245.

---

## 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 IEAAPFxx 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 LNKSTxx 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.

#### 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 IEAAPFxx 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 LNKSTxx 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 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 LNKSTxx or PROGxx (if available on your system) member in 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.

#### 4. Adding IDI.SIDIALPA to the LPALIST

Fault Analyzer modules that must be loaded into the LPA will 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. Failure to do this will result in abend S16D being issued when Fault Analyzer performs analysis.

#### 5. Performing IPL with CLPA or running IDICZSVC



## Making Fault Analyzer modules available

It is necessary to re-IPL your system with CLPA as the Fault Analyzer installation has added SVC modules to LPA in data set IDI.SIDIALPA, and, optionally, in data set IDI.SIDILPA1.

For the initial install, or any time later when module IDICSVCR in IDI.SIDIALPA is updated, you can do the following if an IPL can not 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 will dynamically install the Fault Analyzer SVC 109 ESR code 53 to your system.

### 6. Adding IDI.SIDILPA1 to the LPALIST

Fault Analyzer modules that can optionally be loaded into the LPA will reside in the target library, IDI.SIDILPA1. If this library is 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.SIDILPA1 to your concatenated LPALIST via the LPALSTxx member in your SYS1.PARMLIB. All modules in IDI.SIDILPA1 are 31-bit and are therefore automatically loaded into ELPA above the line.

#### Notes:

- 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.SIDILPA1 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 365.

---

## Defining program control access to Fault Analyzer programs

If security server program control has been 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:

```
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 54).
- The IDIUTIL batch utility SetFaultPrefix and SetMaxFaultEntries functions (for details, see "IDIUTIL control statements" on page 354).



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 261).

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 for users in the default group PAYROLL only:

```
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 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 used. For details, see “Detail” on page 465.

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 will always be 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.

---

### Japanese feature note

For the Japanese feature of Fault Analyzer, an additional 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 used. For details, see “Detail” on page 465.

## Setting up the message and abend code explanation repository

Ensure that all users have READ access to the data set defaulted to, or specified using, the IDIVSJPN DDname.

End of Japanese feature note

### Korean feature note

For the Korean feature of Fault Analyzer, an additional repository must be set up using sample job IDISVKOR 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 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.

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 will then attempt 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 will be generated. This 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 will still be used if the user has ALTER access to the TDUMP data set profile but no XFACILIT access.

Fault Analyzer will try to use SDUMP as the preferred dump type, and only if the necessary SDUMP access authorization is not available for the abending user ID, will the TDUMP access authorization be 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 it's 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 will only use SDUMP in the recovery fault recording process if the user ID under which the abend occurred has been 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.

If by chance a fault entry creation has an exception requiring an RFR dump, then Fault Analyzer will only create and link 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 will provide 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 requestor 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

## Managing recovery fault recording data set access

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 will still be 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 (==IDISRFR)” on page 246.

**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&SYSCclone., 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 will implicitly have 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.

If by chance a fault entry creation has an exception requiring an RFR dump, then Fault Analyzer will only create and link a TDUMP to the fault entry if the user has ALTER access to the appropriate XFACILIT IDIRFR\_TDUMP\_HLQ profile, or ALTER access to the TDUMP data set profile.

If a user doing problem analysis has read or delete access to a fault entry, and the fault entry has a TDUMP linked to it (the fault entry was created by a recovery

fault recording exception), then Fault Analyzer will provide the equivalent access to the TDUMP as an extension to the fault entry. 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 can be modified or expanded on by an installation as required.

**Note:** If the IDISRFR USERMOD has been installed to change the default high-level qualifier, then IDIRFRHQ in the following should be substituted by the actual high-level qualifier used.

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)
```



---

## Chapter 14. Using the Fault Analyzer IDIS subsystem

Fault Analyzer utilizes 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  
ID10082E 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 is primarily intended for CICS system dumps.
- 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 235. For details, see “Caching of history file \$\$INDEX data” on page 234.
- 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 235. 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 235. For details, see “Fast Exclude options processing” on page 270.
- Provide recovery fault recording support. (For general information about recovery fault recording, see “Recovery fault recording” on page 28.)
- Provide improved performance of message and abend code explanation retrieval for the Fault Analyzer ISPF interface LOOKUP command by caching the information in storage.
- 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.

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 subsystem will 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 utilize this feature.

If the XCF should become unavailable, then IDIS subsystem processing will continue to perform all functions, although possibly with reduced performance as a result.

---

### Caching of history file \$\$INDEX data

By specifying the PARM='UPDINDEX' parameter in the IDIS subsystem startup JCL (this is the default—see “Starting the IDIS subsystem” on page 235), the IDIS subsystem will 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 about the \$\$INDEX member, see “Special members in the history file data set” on page 9). PDS-format history files are not be managed in the Fault Analyzer IDIS subsystem because their enqueue and parallel update limitations prevent any effective caching of their \$\$INDEX members.

After reading the \$\$INDEX member from a history file, the information is kept in the IDIS subsystem address space during periods of high activity, providing fast access for any requestors of this data since no I/O is required. Both real-time analysis, interactive and batch reanalysis, as well as the Fault Analyzer ISPF interface, will make use of the cached IDIS subsystem data.

Each time the cache is accessed for read or write, the time limit for in-storage retention is reset. This is to ensure that a history file which remains highly active continues to provide fast cache access. This is particularly beneficial to environments, such as CICS, where multiple abends might occur in rapid succession, and often all needing to update the same history file.

Once the time limit for the IDIS subsystem in-storage retention has expired<sup>8</sup> or a request for update of the same history file is pending from another MVS image in the same sysplex<sup>9</sup>, the \$\$INDEX member is written back to the history file and the IDIS subsystem relinquishes control of it.

Apart from the potential to improve performance in general, the UPDINDEX option can be used to ensure that low-priority jobs that share history files with performance sensitive jobs or execution environments, such as IMS and CICS, in a CPU constrained environment, do not cause excessive delays in the Fault Analyzer execution due to serialization of the history file \$\$INDEX member.

If using the UPDINDEX option, then you must ensure that the Fault Analyzer IDIS subsystem has UPDATE access to all history files that it is expected to handle. Once the IDIS subsystem has been called for a history file to which it does not have UPDATE access, or has failed to update a history file for any reason, no

---

8. The time limit for the IDIS subsystem in-storage retention is currently set to 5 minutes.

9. It is recommended that the use of 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 260.



further attempts to manage the \$\$INDEX member of that history file will be made until the IDIS subsystem has been stopped and restarted.

**Note:** For maximum Fault Analyzer performance, the DeferredReport option should also be considered. 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 ...
//IDISSTST EXEC PGM=IDISAMAN,TIME=NOLIMIT,REGION=region-size,PARM='options'
/* (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 236.

**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 234.

This is the default.

### NOUPDINDEX

Specifies that the abending job will perform 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.

This 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 270.

This is the default.

## Starting the IDIS subsystem

### NOFASTEXCLUDE

Disables fast Exclude options processing to revert back to normal Exclude processing by the IDIDA task.

### XCFGRPSUFFIX=c

Provides control over the last character, *c*, of the IDIS subsystem XCF messaging group name, IDISXCF*c*, in order to create an alternative XCF message group. This 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.

### NOXCFGRPSUFFIX

Use the default "M" suffix.

This is the default.

**JAVA** Enables Java analysis. See “IDIS subsystem requirements for Java” on page 237 for more information.

### NOJAVA

Disables Java analysis.

This 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 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.

## 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 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** Is the total number of PDSE history files that will be managed by the IDIS subsystem.

**avg\_num\_entries**

Is 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 353), then this is the number of fault entries that should be included for that history file.

## IDIS subsystem requirements for DB2

If you have more than one version of DB2 installed, or if the DB2 load module library is not in LINKLIST, then 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, as follows:

```
//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 will be 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
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 225). Failure to add IDI.SIDIAUT2 to LINKLIST will prevent Fault Analyzer from loading DLLs which are necessary for the analysis of Java faults. Also, message IDI0158W will be issued by the IDIS subsystem if this situation should occur.

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 will write 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

## Starting the IDIS subsystem

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 will create a subdirectory of the IDIJLIB path, using the &SYSCClone name. This ensures that each image writes to a separate directory.

---

## Stopping the IDIS subsystem

The IDIS subsystem may be stopped and restarted at any time. When the IDIS subsystem is inactive (stopped), all Fault Analyzer processes will continue to run, except for some DB2 catalog data access. There will also be an increase in CPU and I/O usage, compared to having the IDIS subsystem running with `PARM='UPDINDEX'`.

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 will be 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 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
ISPF LIB	IDI.SIDIPLIB
ISPM LIB	IDI.SIDIMLIB
ISPS LIB	IDI.SIDISLIB
ISPT LIB	IDI.SIDITLIB
SYSEXEC	IDI.SIDIEEXEC

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 edit or browse 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 241), the LOOKC command (see “Invoking the LOOKUP command using cursor selection (LOOKC command)” on page 241), and the LOOKUP command (see “LOOKUP” on page 66).

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 will result 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 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 565, which can be modified to suit any specific requirements your installation might have.

---

### Invoking Fault Analyzer using an ISPF 3.4 line command (FA)

As an alternative to the typical invocation of the Fault Analyzer ISPF interface, using an option from an ISPF selection menu, it might at times be more convenient to invoke the Fault Analyzer ISPF interface directly as a line command against an MVS dump data set or a history file on the ISPF option 3.4 Data Set List Utility panel. This has the advantage of being able to perform interactive dump analysis or display fault entries using the selected data set immediately, without first having 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 allocated to the SYSEXEC concatenation of the ISPF user. This exec name can then be entered next to a dump data set or history file name in an ISPF option 3.4 data set list. The exec will invoke Fault Analyzer using the correct ISPF NEWAPPL ID, and will pass the name of the data set as a parameter. The exec will determine if the data set is a dump data set, based on the logical record length being 4160 bytes. If it is not, then it is assumed that the data set is a valid history file.

The FA exec:

## Invoking Fault Analyzer using an ISPF 3.4 line command (FA)

```
/* REXX */
Parse Arg dsn .

/* Determine if dsn is a history file or an MVS dump */
outl. = ''
x = Outtrap('OUTL.',,,"NOCONCAT")
Address TSO "LISTDS " || dsn
x = Outtrap('OFF')
svcdump = 0
If outl.0 > 2 Then
  Do
    lrecl = Word(outl.3,2)
    If lrecl = 4160 Then
      svcdump = 1
  End

/* Invoke the Fault Analyzer ISPF interface */
Address ISPEXEC
If svcdump = 1 Then
  'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) PARM(DSN('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 will work 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 239.

---

## 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 239.



---

### 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. This 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).

**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 is because users, who might make additional changes to these columns, will otherwise not be able to reset the columns to the installation-wide defaults by pressing PF4.

3. Exit from the Fault Analyzer ISPF interface.
4. Copy the *applid*PROF 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 which must be made available to all users in their ISPTLIB concatenation.

As soon as a user has made changes to any of the variables contained in the profile member, it will be 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 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 will be inserted immediately following the JOB card of the generated JCL stream. The skeleton 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
```



---

## Chapter 16. Customize Fault Analyzer by using USERMODs

Some aspects of the Fault Analyzer customization can only be performed via SMP/E USERMODs (described in the following), or by using an IDIOPTLM configuration-options module (for details, see “Configuration-options module IDIOPTLM” on page 454).

---

### Enabling Fault Analyzer to be invoked

In order for Fault Analyzer to be given control when a program abends, one or more invocation exits must be installed.

The following provides information about the installation of the two invocation exits that are provided for non-CICS environments. Refer to Chapter 21, “Customizing the CICS environment,” on page 319 for invocation exits for the CICS environment.

#### Installing the MVS change options/suppress dump exit IDIXDCAP (++)IDITABX)

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 will include 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.

- Re-IPL with CLPA.

#### Enabling the Language Environment abnormal termination exit IDIXCEE

Fault Analyzer provides a Language Environment abnormal termination exit, IDIXCEE, as an additional 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

## Enabling Fault Analyzer to be invoked

```
CEEXAHD      ,User exit header
CEEXART TERMxit=IDIXCEE
CEEXAST      ,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 re-issue 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
```

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, CEEEXTAN, which is installed with sample job IDIXCCEE, is very similar to this exit. If you are going to install both, make sure that you double-check that you have 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.

### Identifying the LE run-time library (++)IDILEDS)

If Fault Analyzer is to be run against abends occurring in a non-LE environment, and LE is not in LINKLIST, and the LE run-time library data set name is not CEE.SCEERUN, then the IDILEDS USERMOD must be applied to identify the LE run-time library data set name (which must be APF authorized). If LE is in LINKLIST, or if the LE run-time data set name is CEE.SCEERUN, then this USERMOD need not be applied.

The USERMOD is applicable to Fault Analyzer real-time execution using the CICS XPCABND global user exit (IDIXCX52 or IDIXCX53) or the IEAVTABX MVS change options/suppress dump exit (IDIXDCAP) only. It is not applicable to the Fault Analyzer ISPF interface.

## Working with applications that use a non-Language Environment run time

To apply the IDILEDS USERMOD, edit and submit the sample job IDILEDS. Refer to the instructions in the sample job for information about changes you may need to make to this job.

**Note:** An IDIOPTLM configuration-options module may be used instead of this USERMOD—for details, see “Configuration-options module IDIOPTLM” on page 454.

### 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 run-time library, the IDISPLI or IDISPLIA USERMOD must be applied. Either USERMOD modifies the ONCODE processing in the PL/I run-time 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 this will prevent Fault Analyzer from being invoked (see “Application-handled error conditions” on page 224). 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)

The IDITABD USERMOD is required for all non-LE batch jobs, for which fault analysis is to occur, unless either a SYSABEND, SYSUDUMP, or SYSMDUMP DD statement is used. This 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, re-IPL with CLPA. (It is not possible to activate this change using the SETPROG command.)

---

### Specifying an alternative parmlib data set for IDICNF00 (++IDISCNF)

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 supplies an SMP/E USERMOD that permits an alternative data set to be specified for the IDICNF00 configuration member.

The specified data set name may contain MVS system symbols.

The sample USERMOD, IDISCNF, is contained in the data set IDI.SIDISAM1. Refer to the comments in the job for more information about the changes you need to make to this USERMOD.

**Note:** An IDIOPTLM configuration-options module may be used instead of this USERMOD—for details, see “Configuration-options module IDIOPTLM” on page 454.

---

### Changing the default recovery fault recording IEATDUMP data set name (++IDISRFR)

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 28.)

The data set name used is obtained from the IDIRFRDS CSECT. This data set name is by default set to:

```
IDIRFRHQ.IDIRFR.&SYSNAME..D&YYMMDD..T&HHMMSS..&JOBNAME.
```

Like the default data set name provided, the data set name specified here should contain MVS system symbols to ensure that each allocated data set is unique.

**Notes:**

1. 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 228 for additional information.
2. With APAR PK89333 installed, Fault Analyzer no longer replaces the IDIRFRDS CSECT name high-level qualifier by the user ID associated with the abending job, if failing to allocate the IDIRFRDS CSECT name.

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.

To change the default recovery fault recording IEATDUMP data set name, a sample USERMOD, IDISRFR, is provided in the IDI.SIDISAM1 data set. Refer to the comments in the job for more information about the changes you need to make to this USERMOD.

**Note:** An IDIOPTLM configuration-options module may be used instead of this USERMOD—for details, see “Configuration-options module IDIOPTLM” on page 454.

---

## Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (++)IDISXCUM)

This USERMOD 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.

When a prior CICS exit program has specified a return code UERCBYP (suppress transaction dump), then Fault Analyzer will by default continue to perform analysis. This behavior is equivalent to the way in which non-CICS abends are handled by the MVS pre-dump exit, IDIXDCAP.

With this USERMOD installed, a prior XDUREQ exit program return code of UERCBYP will stop Fault Analyzer from performing analysis.

To install this USERMOD, edit and submit member IDISXCUM in the IDI.SIDISAM1 data set. Refer to the comments in the job for more information about the changes you need to make to this USERMOD.

**Note:** An IDIOPTLM configuration-options module may be used instead of this USERMOD—for details, see “Configuration-options module IDIOPTLM” on page 454.

---

## 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 will be 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 will include 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 will normally only set the DD name field to DFHRPL when running under CICS, on most other occasions the DD name will be blank, indicating Fault Analyzer will leave 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

## Obtaining load modules from CA-Panexec

to leave the DD name unchanged. If the DD name is updated, then it will be used for the Binder INCLUDE call of the module.

A sample CA-Panexec exit is available as member IDIPANEX in data set IDI.SIDISAM1.

---

## Chapter 17. Setting up history files

Fault Analyzer requires at least one PDS or PDSE data set to be allocated as a fault history file.<sup>10</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*

is the maximum number of concurrent entries that you wish 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 is because of their automatic space management and their 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 may 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.

To allocate one or more history files, edit and submit the sample job IDISHIST. This job will allocate two PDSE data sets named IDI.HIST and IDI.HIST.TEST respectively. All users whose abending jobs will be recorded in these history files will need write access to the data sets.

Refer to the instructions in the sample job for information about changes you may 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 53.

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 will cause Fault Analyzer to abend (for example, system abend SE37).

---

10. 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

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 (SetMaxFaultEntries control statement). See “AUTO-managed PDSE history files” for additional 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 353.

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 260.

---

## AUTO-managed PDSE history files

By default, a newly created PDSE history file is AUTO-managed. Alternatively, this can be controlled using the IDIUTIL batch utility SetMaxFaultEntries control statement, as described in “SETMAXFAULTENTRIES control statement” on page 357.

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, but can be specified using the IDIUTIL batch utility SetMaxFaultEntries control statement. 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 will automatically delete the oldest eligible fault entries, in excess of 25, to provide the space required for writing new fault entries. This generally prevents the allocation of additional 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 35.

A view is essentially a list of history files, that will all be 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 261), 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 253.



The list of history file data set names to make up a view are simply placed into a member in another PDS that has been created specifically to hold the view definitions. This would generally be 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. This allows the comment line to provide a description of the view to assist the users choice.

Figure 117 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 117. Sample view data set member*

In the above example:

**1** This is a comment (asterisk in column 1). Because this comment is on the first line of the view member, it will also be 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 will be displayed simultaneously through the Fault Analyzer ISPF interface when selecting this view. All data set names must start in column 1 and be specified on a single line each. There is no limit on the total number of data set names that can be specified in a single view.

The same symbol substitution rules apply to the data set names specified in a view member, as those which apply to the data set names specified in the IDIVIEWS suboption of the DataSets option—for details, see “Data set name substitution symbols” on page 461.

**4** This is a comment that will simply be ignored by the Fault Analyzer ISPF interface.

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” on page 252) or -Match (see “Specifying an initial fault entry selection criteria” on page 252) 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 will help users recognize the owning group when viewing faults in a composite view display. The fault prefix characters may 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 on page 477. 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 41).

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 will take effect.

Figure 118 shows an example of a view data set member specifying a default column layout.

---

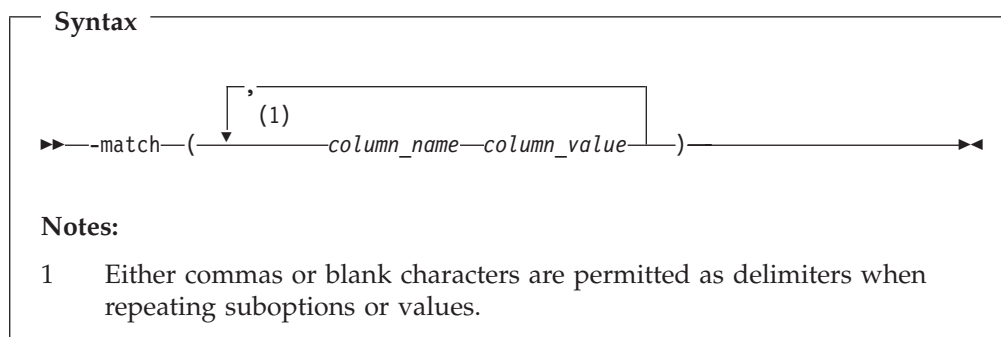
```
* All IMS history files
-HistCols(IMS_Pgm Jobname User_ID System Abend Date      Time      )
SYS1.IMS.HIST
SYS2.IMS.HIST
```

---

Figure 118. 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.



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 which match the specified string will be 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 will take effect.

Figure 119 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 119. 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 will already be 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 will use. All users in the same group will thus have read and delete access to the faults in a history file set up for the group.

This will generally mean that it is better to maintain separate fault history files for different user groups, providing the write (update) access only to the history file their jobs will use. However, this makes it harder for a user to look at the faults across the environment if they are in many different history files. This is where a "view" can be created 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 is only possible when all history files in a view are accessible from the same MVS system.

## Managing history files across MVS systems without shared DASD

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 is described in “Automated implementation.”
- The second is an implementation which can be used to transmit fault entries "on demand" during interactive reanalysis. This is described in “On demand implementation” on page 258.

### Automated implementation

The following is an example that shows how fault entries created on MVS system 'A' can be automatically copied to MVS system 'B' for viewing or reanalysis using the Fault Analyzer ISPF interface when the two systems do not share any DASD (if they did share DASD, they could simply use a common history file). This is accomplished by using a Notification user exit<sup>11</sup> on system 'A' that submits a TSO batch job to transmit fault entries as PDS members to a dedicated user ID on system 'B'. On system 'B', a continually running batch TSO job receives fault entries into a staging data set and then calls the IDIUTIL batch utility<sup>12</sup> to import the fault entry into a local history file. Figure 120 on page 255 illustrates this concept.

---

11. See Chapter 31, “Customizing Fault Analyzer by using user exits,” on page 369 for information about user exits in general and “Notification user exit” on page 405 for information about the Notification user exit specifically.

12. See Chapter 27, “Managing history files (IDIUTIL utility),” on page 353 for information about the IDIUTIL batch utility.

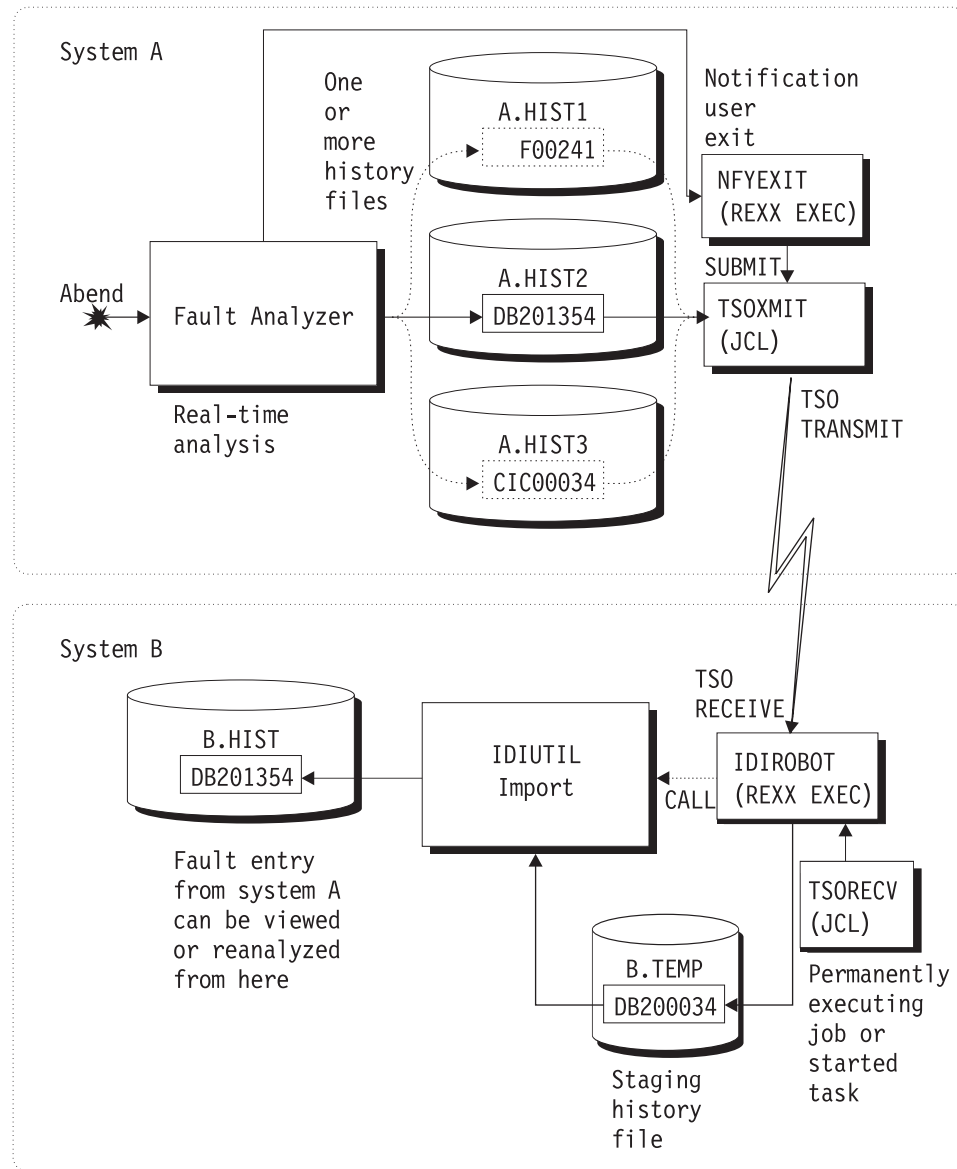


Figure 120. Fault entry import via TSO XMIT/RECEIVE

Figure 121 on page 256 shows the sample Notification REXX user exit, NFYEXIT. It is available in softcopy format as member IDISXNFY in data set IDI.SIDISAM1.

---

```

nodeid   = 'MVS' /* <--- verify/change */
userid   = 'IDIROBOT' /* <--- verify/change */
jobcard  = '//NOTIFY JOB MSGCLASS=Z' /* <--- verify/change */
/*****
/* #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_IDHIST = "MY.HISTFILE" then 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("ENV.FAULT_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)"
say 'Fault entry' ENV.FAULT_ID 'sent to' nodeid '.'userid
exit 0

```

---

Figure 121. Sample Notification user exit (NFYEXIT) 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.
- 2** The 'userid' variable is the user ID for which fault entries will be received on the target system. This should be a user ID used solely for the purpose of receiving fault entries.
- 3** Ensure that the job card adheres to local standards.
- 4** You can add checks here to see if a fault is eligible to be sent to another system. The example shows how the user ID or history file name could be used, but the checks might be performed on any fields in the ENV or NFY data areas.

The exit in Figure 121 can be invoked for any faults occurring on system 'A' by adding the options:

```

DataSets(IDIEXEC(exec.lib))
Exits(NOTIFY(REXX(NFYEXIT)))

```

to the IDICNF00 config member, where *exec.lib* is your REXX EXEC PDS(E) data set.

Figure 122 on page 257 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 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 = outtrap('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
end

```

---

Figure 122. Sample TSO receive REXX exec (IDIROBOT)

Notes:

- 4** This is the name of the target history file in which the received fault entries are placed. See **8** if you would like to select the history file based on where the fault originally occurred.

## Managing history files across MVS systems without shared DASD

- 5** This is a staging history file that is used for the TSO receive command and from which fault entries are imported into the target history file and subsequently 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. It is recommended that this data set be made a PDSE for its automatic space reclamation capabilities.
- 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 before entering the WAIT.
- 8** Although the sample exec uses only a single target history file for all received fault entries, it would be possible to assign a target history file based on 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'), or the fault ID itself (in variable 'faultid').

Figure 123 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 123. 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 will execute 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



## Managing history files across MVS systems without shared DASD

“Automated implementation” on page 254, this implementation uses a Formatting user exit to perform the same function, but only as required from within the interactive reanalysis report.

Figure 124 shows the sample Formatting REXX user exit, IDIXMIT. It is available in softcopy format as member IDIXMIT in data set IDL.SIDISAM1.

---

```

nodeid  = 'MVS' /* <--- verify/change */ 1
userid  = 'IDIROBOT' /* <--- verify/change */ 2
jobcard = '//NOTIFY JOB MSGCLASS=Z' /* <--- verify/change */ 3
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('ENV.FAULT_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)"
/*****/ 4
/* #Optionally, update the user title field to show that the fault
/* has been sent.
/* For example:
/* ENV.USER_TITLE = "Sent to" nodeid "on" DATE()".
/*****/
/*****/ 5
/* #Optionally, tell the user that the request has been processed
/* by uncommenting the following IDIWRITE calls:
/* "IDIWRITE '<p>History file' ENV.IDIHIST "fault ID" ENV.FAULT_ID,
/* "sent to" nodeid "."userid"."
/* "IDIWRITE '<p>Press PF3 to return to the interactive reanalysis",
/* "report."
/*****/

```

---

Figure 124. 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 might be an automated receiver (as discussed in “Automated implementation” on page 254, or any other node at which the receive and import will be done manually.
- 2 The 'userid' variable is the user ID for which fault entries will be received on the target system.  
  
This might be an automated receiver (as discussed in “Automated implementation” on page 254, or any other user ID for which the receive and import will be done manually.
- 3 Ensure that the job card adheres to local standards.

## Managing history files across MVS systems without shared DASD

- 4** Uncommenting this code provides an optional method for identifying the fault entry as having been sent by providing the date on which this occurred, and the system to which it was sent, in the user title field.
- 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 254, or it might be any user ID and node for which the receive, and subsequent import, will be done manually.

### Using the on demand implementation

While analyzing a fault entry, enter the command

```
EXEC IDIXMIT
```

on the command line.

The IDIXMIT name may 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 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. This should not be done with Fault Analyzer history files, as the common data set name will cause 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 &SYSCZONE 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 utilize 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 250.

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 all systems must be 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 will force 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 209 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 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 4 “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.

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 234.

---

## 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 will 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.

## 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

## Managing history file fault entry access

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 give additional permission via XFACILIT, not restriction. For example, if a user can not 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 associated with the current user (in case of creating a new fault entry), or the security server GROUP and USER ID associated with a fault entry (in the case of viewing, updating, or deleting an existing fault entry).

The security server GROUP and USER ID associated with a fault entry are those which were current for the user who initially created the fault entry.

*hist-dsn* is 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 will check 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 will be performed.

The XFACILIT class profile access levels used by Fault Analyzer translate to Fault Analyzer actions as follows:

*Table 10. Fault Analyzer XFACILIT profile access levels*

Fault Analyzer action	Required XFACILIT profile access level
Read.  For example, viewing the saved report or performing reanalysis.	READ
Write or create.  For example, updating user notes in an existing fault entry, or creating a new fault entry in a history file.	UPDATE or CONTROL
Delete.  This applies to explicit deletions using the D (or DD) line command only.	ALTER

**XFACILIT implementation example 1**

Given a history file named IDI.COMMON.HIST, then by defining

- UACC(NONE) for the IDI.COMMON.HIST data set profile
- UACC(UPDATE) for the IDIHIST\_GROUP\_DSN.*group*.IDI.COMMON.HIST XFACILIT class profile, repeated for each instance of *group*

the resulting access to the IDI.COMMON.HIST history file will be such that

- Faults are accessible through Fault Analyzer only
- Users can view or reanalyze fault entries created by themselves or anyone else in the same security server default group as them

**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:

```
SETROPTS 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)
SETROPTS GLOBAL(XFACILIT) REFRESH
```

The global access table allows &RACUID and &RACGPID and so reduce the administrative effort.

**Notes:**

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:

```
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 will 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 will 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(*)
```

## Managing history file fault entry access

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 will still be done by Fault Analyzer according to the SetMaxFaultEntries 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 will be called. If the IDIXFXIT exit is available, then it might ultimately grant the fault entry access. This might be useful if an installation has previously employed a different access authorization scheme and wishes 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 will be loaded using the Language Environment CEELoad service, and if LE conforming, should not contain a C or PL/I main() function.

**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 11. 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 fault entry creator.
Parameter 3	Varying	Security server default group ID of fault entry creator.
Parameter 4	Varying	The name of the job which created the fault entry.
Parameter 5	Varying	The history file data set name containing the fault entry.
Parameter 6	Varying	The fault entry ID.

**Return specifications:** On return from IDIXFXIT, the contents of registers must be as follows:

Register	Contents
0-1	Undefined.
2-14	Unchanged.
15	Return code:
0	Access not granted.
1	Read access granted.
2	Update access granted.
3	Delete access granted.

**Example (C):** The following is an example of an IDIXFXIT exit written in C.

```
int IDIXFXIT(char *action, char *userid, char *group, char *jobnm,
             char *histDSN, char *member) {
    if (strcmp("MY.HIST", histDSN) == 0) return 2; /* Update access ok */
    else return 0 ;                               /* No access */
}
```





---

## Chapter 18. Setting and changing default options for the site

The creation of a parmlib member, IDICNFxx, 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 IDICNFxx

Default options for the site are held in the parmlib member IDICNFxx, where xx is one of the following:

- 00
- A value matching the current MVS symbol, &SYSCZONE.  
&SYSCZONE is a standard MVS system symbol that may represent a unique 1 or 2-character system ID. Refer to *MVS Initialization and Tuning Reference* for information about the MVS &SYSCZONE symbol.  
Using an IDICNF&SYSCZONE 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&SYSCZONE member name. If this is not found in any parmlib data set, then a search is made for an IDICNF00 member.

The IDICNFxx member may 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 wish to place the IDICNFxx member in the logical parmlib concatenation, a USERMOD can be applied to Fault Analyzer that will permit the specification of an alternative partitioned data set name. See "Specifying an alternative parmlib data set for IDICNF00 (++)IDISCNF" on page 246 for details.

If the IDICNFxx member does not exist, then Fault Analyzer uses the product-supplied default options. In real-time, message IDI0018W is issued.

The following is an example of an IDICNFxx parmlib member:

---

```

/*-----*/
/* 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 */
  IDILCOB (APP2.LISTINGS)     /* OS/VS COBOL compiler listings */
  IDILPLI (MY.LISTING.PLI    ) /* PL/I compiler listings */
  IDILANGX(MY.IDILANGX)       /* IDILANGX files */
  IDIADATA(MY.SYSADATA)       /* SYSADATA files */
  IDIMAPS (IDI.SIDIMAPS)      /* Data area models */
)

```

---

*Figure 125. 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 through 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 the following DDnames:  
 IDIBOOKS  
 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 page 458).

You can change the parmlib member whenever you need to. For example, you may change the parmlib member to exclude additional 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 89, and “Interactive reanalysis options” on page 97.

The individual options are explained in “Option descriptions” on page 455.

**Note:** If a user-options module is used, it may replace the IDICNFxx default options. For details, refer to “User-options module IDICNFUM” on page 453.

The following 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 wish 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 may 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 468) 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 (this is 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 will be excluded from analysis. In this case, you will get no analysis report, and no fault entry is written to the history file.

### Notes:

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 will be 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:

## Controlling which jobs are analyzed with Exclude processing

```
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 */
```

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 is the default), then all Include and Exclude options specifications from the IDICNF00 parmlib member are cached in the IDIS subsystem. This allows a Fault Analyzer invocation exit to 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 re-read.

To disable fast Exclude options processing, specify the IDIS subsystem PARM='NOFASTEXCLUDE' option.

---

## Controlling report detail

The Detail option described 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 16 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 (described on page 481) 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 will be written.

When choosing a minidump size limit, the space 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 (described on page 458) 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 382.

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, the `Quiet` option may be used.

For information about the `Quiet` option, see “Quiet” on page 492.

---

## Customizing the Fault Entry List display

If an installation-wide change to the information shown on the Fault Entry List display is required, this can be achieved by using the `HistCols` option to specify the particular columns of information to be displayed in the order desired. This will provide users with a common base from which they can make further changes if they wish.

For information about the `HistCols` option, see “HistCols” on page 477.

---

## Specifying the cultural environment

The `Locale` option (described on page 480) lets you specify the locale to be used in order to enable cultural environment-dependent presentation.



---

## Chapter 19. Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products

This chapter describes the minimal steps required to prepare your programs for use with IBM Problem Determination Tools products. For more detailed information, refer to 'Part 2. Preparing your program for debugging' of the *Debug Tool for z/OS User's Guide*, 'Part 2. Fault Analyzer Installation and Administration' of the *Fault Analyzer for z/OS User's Guide*, or 'Appendix B. Creating side files using CAZLANGX' of the *Application Performance Analyzer for z/OS User's Guide*.

The purpose of this chapter is to provide instructions for a single compile method for organizations that are using some combination of Debug Tool for z/OS, Fault Analyzer for z/OS, and Application Performance Analyzer for z/OS. If your enterprise is only using Debug Tool for z/OS, you can alternatively refer to 'Part 2. Preparing your program for debugging' of the *Debug Tool for z/OS User's Guide*. If your enterprise is only using Fault Analyzer for z/OS, alternatively refer to 'Part 2. Fault Analyzer Installation and Administration' of the *Fault Analyzer for z/OS User's Guide*. If your enterprise is only using Application Performance Analyzer for z/OS, alternatively refer to 'Appendix B. Creating side files using CAZLANGX' of the *Application Performance Analyzer for z/OS User's Guide*.

Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS are designed to use load modules and other files produced by IBM compilers. You must compile your programs with certain compiler options so that they produce load modules and files that these products can use.

This chapter uses the term 'source information files' to refer to the types of files that are used by Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. The different kinds of source information files that are the subject of discussion in this chapter include:

- SYSDEBUG files
- LANGX files
- Compiler listings
- DWARF files
- Expanded source files

Be aware that different compilers generate different kinds of source information files. If you use more than one compiler, you might have more than one type of source information library.

When you compile your programs with the compiler options described in this chapter, you can use the load modules and source information files created by the compilers as follows:

- Prepare the module for debugging (if you are using Debug Tool for z/OS). Debug Tool for z/OS lets you work with program statements and variables.

When a program is compiled with the right options, the module that is produced by the compiler can be debugged and a source information file, which contains program statements, can be produced. When you use Debug Tool for z/OS to debug a program, Debug Tool for z/OS uses the source information file to display the program source statements in the source window.

Depending on the source language and compiler used, either the load module, source information file, or DWARF file contains information about statements and variables, such as offsets and lengths, and contains information that allows the debugger to locate statements and variables in storage. If you do not compile with the correct compile options, debugging is limited to something called 'disassembly' mode, where machine code is displayed, but no source statements or variables.

- Utilize Fault Analyzer for z/OS to automatically pinpoint the source statement that caused an abend, and can show you the values of variables in your programs at the time of an abend.
- Utilize Application Performance Analyzer for z/OS to show you precisely which program statements are utilizing the most CPU time and wait time, to give you information you need to tune your applications.

---

## Updating your build process

If someone recently installed one or more of the IBM Problem Determination Tools products on your system, the program build processes might not have been updated yet. Updating the build processes is an important and necessary part of implementing the IBM Problem Determination Tools products.

In many organizations, there is clear ownership of these build processes. In other organizations, it might not be obvious who should make the changes. Many organizations use standard compile processes or PROCs that your system administrators maintain and have updated to prepare programs for the IBM Problem Determination Tools products. If this is the case, find out what processes have been made available and how to use them. In other organizations, each developer maintains their own compile JCL or PROCs to compile programs. If this is the case, update your own compile JCL to prepare your programs for the IBM Problem Determination Tools products as described below.

Start by researching what is required for each compiler individually. For example, the changes required for Enterprise COBOL for z/OS, Enterprise PL/I for z/OS, C/C++ and Assembler are all slightly different.

In general, there are three changes that might be needed to compiler JCL to produce programs that can be used by the IBM Problem Determination Tools:

1. Specify compiler options required by the IBM Problem Determination Tools. For example, in the case of Enterprise COBOL for z/OS, a TEST options is needed.
2. Code the JCL to produce and save the source information files that the IBM Problem Determination Tools products need. Newer compilers can generate the required source information files directly. Some older compilers require an additional step in the compile job to run a special utility program that produces the needed file.
3. In certain environments, it is advantageous to include a special Debug Tool for z/OS module into the application load module during the link edit step. In most cases this is optional, but it can simplify starting Debug Tool for z/OS for certain types of programs. For certain older compilers running in certain environments, you must include a special module to enable Debug Tool for z/OS.



---

## Updating your promotion process

Typically, when a program is tested, program load modules are promoted through different stages before reaching production. For example, when a new program is compiled for the first time, it might be placed into a test load library. After unit testing is completed, perhaps the compiled program is promoted to a quality assurance environment. And eventually, it is promoted into production. On your system, you might know these stages by different names, such as:

- Unit test
- System test
- Model office

Consider whether you want the ability to use Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS throughout your programs' life cycle. Even if you do not plan to use Debug Tool for z/OS with production programs, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS are very useful in those stages. To enable the IBM Problem Determination Tools products at each stage, update your promotion processes to retain the source information files. Promotions can be accomplished by performing a recompile, a copy, or a move. Perform the same steps with your source information files that you perform with your load modules or object modules. For each load library or object library, you should have a corresponding set of source information libraries. Whenever you promote a load module or object module, you should promote the source information file as well. This ensures that the source information file is available for Fault Analyzer and Application Performance Analyzer, and you can continue to take advantage of the IBM Problem Determination Tools products at all stages of your program's life cycle.

---

## Preparing your programs

Each compiler produces different kinds of source information files, and each of the IBM Problem Determination Tools products reads different kinds of files. It can be time-consuming to research all the different combinations, but for each compiler, there is a suggested method described below. If you use the suggested methods, then your programs will be ready to take full advantage of the IBM Problem Determination Tools products.

- "Enterprise COBOL for z/OS Version 4 programs"
- "Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs" on page 278
- "COBOL for MVS and VM programs" on page 280
- "VS COBOL II programs" on page 282
- "OS/VS COBOL programs" on page 284
- "Enterprise PL/I Version 3.7 and later programs" on page 285
- "Enterprise PL/I Version 3.5 and Version 3.6 programs" on page 288
- "Enterprise PL/I Version 3.4 and earlier programs" on page 291
- "PL/I for MVS and VM and OS PL/I programs" on page 294
- "z/OS XL C and C++ programs" on page 296
- "Assembler programs" on page 300

### Enterprise COBOL for z/OS Version 4 programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 4 programs for use with the IBM Problem

Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant runtime overhead.

*Table 12. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for Enterprise COBOL for z/OS Version 4*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(NOHOOK, SEPARATE, EJPD), LIST, MAP, SOURCE, NONNUMBER, XREF(SHORT)	SYSDEBUG	Yes	Suggested for production and test		
NOTEST, LIST, MAP, SOURCE, NONNUMBER, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, NUMBER, XREF(SHORT)		Yes	N/A	Supported	N/A
LIST, MAP, SOURCE, NONNUMBER, XREF(SHORT)	LANGX file	Yes	N/A	Supported	Supported

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

## Preparing Enterprise COBOL for z/OS Version 4 programs

Perform the following steps for compiling your Enterprise COBOL for z/OS Version 4 programs using the compiler options suggested in Table 12:

1. Create libraries (PDSE is suggested unless PDS is required in your organization) for SYSDEBUG files. Create one or more SYSDEBUG libraries for each environment, such as test, production, and so on.
2. Create a corresponding SYSDEBUG library for each load library. Specify LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of 1recl < 32K).
3. For all programs in both test and production environments, specify the following compiler options:  
TEST(NOHOOK,SEPARATE,EJPD),LIST,MAP,SOURCE,NONNUMBER,XREF(SHORT).

The TEST compiler option is required if you plan to use Debug Tool for z/OS to debug a program. The TEST option is optional if you plan to use Fault Analyzer for z/OS or Application Performance Analyzer for z/OS.

The SEPARATE sub-option produces a SYSDEBUG file.

NOHOOK and SEPARATE produce a production-ready module that can still be debugged.

If the OPT option is also used, EJPD might reduce optimization but enables the debugger's JUMPTO and GOTO commands. These commands are disabled when OPT and NOEJPD are both used.

4. When the TEST option is not used, save the compiler listing in a file, or use the xxxLANGX utility program to create a LANGX file. Equivalent xxxLANGX utilities are available in Debug Tool for z/OS as EQALANGX, in Fault Analyzer for z/OS as IDILANGX and in Application Performance Analyzer for z/OS as CAZLANGX. Fault Analyzer for z/OS and Application Performance Analyzer for z/OS can use compiler listings and LANGX files to provide source-level support.
5. The LIST, MAP, SOURCE, and XREF options are needed only if a compiler listing or a LANGX file will be used to provide source information to Fault Analyzer for z/OS or Application Performance Analyzer for z/OS. If a SYSDEBUG file will be used with these products or if you will not be using Fault Analyzer for z/OS or Application Performance Analyzer for z/OS, the LIST, MAP, SOURCE, and XREF options are optional.
6. The NONUMBER compiler option is needed only if a compiler listing file will be used to provide source information to Application Performance Analyzer for z/OS. If a SYSDEBUG file will be used with Application Performance Analyzer for z/OS, or if you will not be using Application Performance Analyzer for z/OS, the NONUMBER option is optional.
7. Code a SYSDEBUG DD in the JCL of the compiler step:  

```
//SYSDEBUG DD DSN= SYSDEBUG.pds (pgmname) ,DISP=SHR
```

Save the SYSDEBUG file produced by the compiler in the SYSDEBUG library and specify a member name that is equal to the program name of your application program. This is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.

8. Modify the promotion process to promote SYSDEBUG files. When a load module is promoted, for example from test to production, promote the corresponding SYSDEBUG file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the SYSDEBUG file that you perform with the module during promotion.
9. Optionally, include a Debug Tool Language Environment (LE) exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

### **Sample JCL for compiling Enterprise COBOL for z/OS Version 4 programs**

A sample job for compiling an Enterprise COBOL for z/OS Version 4 program for use with the IBM Problem Determination Tools products is available as member IDISJ001 in data set IDI.SIDISAM1. This is a generic sample, and might not meet all your requirements to generate your modules.

Notice that the TEST compiler option is specified. Code the correct sub-options of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also notice that a SYSDEBUG DD statement has been coded. This is the source information file that the compiler produces. It refers to a SYSDEBUG library that is a PDS or PDSE. The member name must be the same as the program name.

For Enterprise COBOL for z/OS, these are the only required changes.

However, there is an optional change in the linkage editor step. The example below shows that a special Language Environment exit module is included in the application load module. Although this is not required, it enables the use of Debug Tool panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start Debug Tool, this is one way to enable it. If you do not plan to use Debug Tool panel 6, then do not include an exit module.

## Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant runtime overhead.

*Table 13. Examples of compiler options and source information files supported by IBM Problem Determination tools products for Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(NONE, SYM, SEPARATE), LIST, MAP, SOURCE, NONNUMBER, XREF(SHORT)	SYSDEBUG	Yes	Suggested for production and test		
NOTEST, LIST, MAP, SOURCE, NONNUMBER, NOOPT, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, XREF(SHORT), NUMBER		Yes	N/A	Supported	N/A
LIST, MAP, SOURCE, NONNUMBER, XREF(SHORT)	LANGX file	Yes	N/A	Supported	Supported

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

### Preparing Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs

Perform the following steps for compiling your Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs using the compiler options suggested in Table 13:

1. Create libraries (PDSE is suggested unless PDS is required in your organization) for SYSDEBUG files. Allocate one or more SYSDEBUG libraries for each environment, such as test, production, and so on.
2. Create a corresponding SYSDEBUG library for each load library. Specify `LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K)`.
3. For all programs in both test and production environments, use `TEST(NONE,SYM,SEPARATE),LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)`.  
`TEST` is required by Debug Tool for z/OS.  
 The `SEPARATE` sub-option produces a SYSDEBUG file. Specifying `NONE` with `SEPARATE` produces a production-ready module that can still be debugged.  
 If `OPTIMIZE` is specified, the debugger `JUMPTO` and `GOTO` commands are disabled. These commands are enabled when `NOOPTIMIZE` is specified.
4. The `LIST`, `MAP`, `SOURCE`, and `XREF` options are needed only if a compiler listing or a `LANGX` file will be used to provide source information to Fault Analyzer for z/OS or Application Performance Analyzer for z/OS. If a SYSDEBUG file will be used with these products, or if you will not be using Fault Analyzer for z/OS or Application Performance Analyzer for z/OS, the `LIST`, `MAP`, `SOURCE`, and `XREF` options are optional.
5. The `NONUMBER` compiler option is needed only if a compiler listing file will be used to provide source information to Application Performance Analyzer for z/OS. If a SYSDEBUG file will be used with Application Performance Analyzer for z/OS, or if you will not be using Application Performance Analyzer for z/OS, the `NONUMBER` option is optional.
6. Code a SYSDEBUG DD in the JCL of the compiler step.  

```
//SYSDEBUG DD DSN= SYSDEBUG.pds(pgmname),DISP=SHR
```

Save the SYSDEBUG file produced by the compiler in the SYSDEBUG library and specify a member name that is equal to the program name of your application program. This is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.

7. Modify the promotion process to promote SYSDEBUG files. When a load module is promoted, for example from test to production, promote the corresponding SYSDEBUG file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the SYSDEBUG file that you perform with the module during promotion.
8. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module `EQADBCXT` for batch programs (including IMS batch), `EQADICXT` for IMS/TM programs and `EQADDCXT` for DB2 stored procedures. Do not include the exit module for CICS programs.

### **Sample JCL for compiling Enterprise COBOL for z/OS Version 3 programs**

A sample job for compiling an Enterprise COBOL for z/OS Version 3 program for use with the IBM Problem Determination Tools products is available as member `IDISJ002` in data set `IDI.SIDISAM1`. This is a generic sample, and might not meet all your requirements.

Notice that a TEST option is specified. Code the correct sub-option of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also notice that a SYSDEBUG DD statement has been coded. This is the source information file that the compiler produces. It refers to a SYSDEBUG library that is a PDS or PDSE. The member name must be the same as the program name.

For Enterprise COBOL for z/OS, these are the only required changes.

However, there is an optional change in the linkage editor step. The example below shows that a special Language Environment exit module is included in the application load module. Although this is not required, it enables the use of Debug Tool panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start Debug Tool, this is one way to enable it. If you do not plan to use Debug Tool panel 6, then do not include an exit module.

## COBOL for MVS and VM programs

The following table shows various compiler options that can be used to prepare COBOL for MVS and VM programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 14. Examples of compiler options and source information files supported by IBM Problem Determination tools products for COBOL for MVS and VM*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(ALL, SYM), LIST, MAP, SOURCE, NOOPT, NONUMBER, XREF(SHORT)	Compiler listing	No	Suggested for test. (Using Debug Tool in production for this compiler is not suggested.)		
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)		Yes	N/A	Suggested for production	
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	LANGX file	Yes	N/A	Supported	Supported

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

## Preparing COBOL for MVS and VM programs

Perform the following steps for compiling your COBOL for MVS and VM programs:



1. Create libraries (PDSE is suggested unless PDS is required in your organization) for compiler listing files. Allocate one or more compiler listing libraries for each environment, such as test and production.
2. Create a corresponding listing library for each load library. Specify  
LRECL=133,RECFM=FBA,BLKSIZE=(multiple of 1recl < 32K).
3. For all programs, such as batch, CICS, and IMS:
  - In test environments, specify compiler options  
TEST(ALL,SYM),NOOPT,LIST,MAP,SOURCE,NONUMBER,XREF(SHORT) to create a module that can be used with Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.  
TEST is required for Debug Tool for z/OS.  
The ALL sub-option adds debug hooks, which will add some run-time overhead.  
SYM stores symbolics data required by Debug Tool for z/OS into the module, which can make it significantly larger.  
The other options format the compiler listing as required by Debug Tool for z/OS, Fault Analyzer for z/OS, and Application Performance Analyzer for z/OS.
  - In production environments, specify compiler options  
NOTEST,LIST,MAP,SOURCE,NONUMBER,XREF(SHORT) to create a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not Debug Tool for z/OS). Specify OPTIMIZE if preferred.  
NOTEST disables source level debugging with Debug Tool, but can provide better performance and smaller module size.  
The other options (except OPTIMIZE) format the compiler listing as required by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
4. Modify the SYSPRINT DD in the JCL of the compiler step to refer to a file.  
//SYSPRINT DD DSN= compiler.listing.pds(pgmname),DISP=SHR  
  
Save the compiler listing in a file in the compiler listing library and specify a member name that is equal to the program name of your application program. This is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
5. Modify the promotion process to promote compiler listing files. When a load module is promoted, for example, from test to production, promote the corresponding compiler listing file or files. A promotion can be a recompile, a copy, or a move. Perform the same steps with the compiler listing file that you perform with the module during promotion.
6. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

### **Sample JCL for compiling COBOL for MVS and VM programs**

A sample job for compiling a COBOL for MVS and VM program for use with the IBM Problem Determination Tools products is available as member IDISJ003 in data set IDI.SIDISAM1. This is a generic sample, and might not meet all your requirements.

Notice that a TEST option is specified. Code the correct sub-options of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also notice that the SYSPRINT DD refers to a permanent file. This is the source information file that the compiler produces. It refers to a listing library that is a PDS or PDSE. The member name must be the same as the program name. For COBOL for MVS and VM, these are the only required changes.

However, there is an optional change in the linkage editor step. The example below shows that a special Language Environment exit module is included in the application load module. Although this is not required, it enables the use of Debug Tool panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start Debug Tool, this is one way to enable it. If you do not plan to use Debug Tool panel 6, then do not include an exit module.

## VS COBOL II programs

If you are currently using the TEST option to compile your programs, consider using NOTEST. Using NOTEST allows you to take advantage of Debug Tool for z/OS functionality that is not available when compiling with the TEST option. Examples of Debug Tool for z/OS functions that are available when compiling with the NOTEST option include the automonitor feature and using AT ENTRY *program name* breakpoints. Compiling with NOTEST also allows you to generate a module that can be debugged but does not incur additional overhead when running without the debugger.

The following table shows various compiler options that can be used to prepare VS COBOL II programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 15. Examples of compiler options and source information files supported by Problem Determination tools products for VS COBOL II*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
NOTEST, LIST, MAP, SOURCE, XREF, NONUMBER, NOOFFSET	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, XREF, NUMBER		Yes	N/A	Supported	N/A
NOTEST, LIST, MAP, NOOPT, SOURCE, XREF, NONUMBER	LANGX file	Yes	Suggested for production and test		

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.



## Preparing VS COBOL II programs

Perform the following steps for compiling your VS COBOL II programs using the compiler options suggested in Table 15 on page 282:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify `LRECL=1562` or greater, `RECFM=VB`, `BLKSIZE= 1recl+4` to 32k.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, compile with `NOTEST`, `LIST`, `MAP`, `NOOPT`, `SOURCE`, `XREF`, `NONUMBER` compiler options.
4. Modify the `SYSPRINT` DD in the compiler step to refer to a file. It can be either a permanent or temporary file. This will be the input to the `xxxLANGX` utility.
5. Add a step after the compiler step to run the Problem Determination tools `xxxLANGX` utility. This utility program reads the compiler listing and generates a LANGX file. This is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library and specify a member name that is equal to the program name of your application program. Equivalent `xxxLANGX` utilities are available in Debug Tool for z/OS as `EQALANGX`, in Fault Analyzer for z/OS as `IDILANGX` and in Application Performance Analyzer for z/OS as `CAZLANGX`.
6. If the module is linked with Language Environment services, optionally include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable the Debug Tool panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module `EQADBCXT` for batch programs (including IMS batch), `EQADICXT` for IMS/TM programs and `EQADDCXT` for DB2 stored procedures. Do not include the exit module for CICS programs or if the module is not linked with Language Environment services (it is linked with COBOL II runtime services).
7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.

## Sample JCL for compiling VS COBOL II programs

A sample job for compiling a VS COBOL II program for use with the IBM Problem Determination Tools products is available as member `IDISJ004` in data set `IDI.SIDISAM1`. This is a generic sample, and might not meet all your requirements.

Notice the compiler options used and notice that the compiler listing is passed to an added step that generates a LANGX file. The compiler listing can be stored in a permanent file or can be passed in a temporary file. For VS COBOL II, these are the only required changes.

However, there is an optional change in the linkage editor step. The following example includes a special Language Environment exit module in the application load module. Although this is not required, it enables the use of Debug Tool panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start Debug Tool, this is one way to enable it. If you do not plan to use Debug Tool panel 6, then do not include an exit module. Do not include the

exit module for CICS programs or if the module is not linked with Language Environment services (it is linked with COBOL II runtime services).

## OS/VS COBOL programs

The following table shows various compiler options that can be used to prepare OS/VS COBOL programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 16. Examples of compiler options and source information files supported by Problem Determination tools products for OS/VS COBOL*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
DMAP, NOCLIST, NOLST, PMAP, SOURCE, VERB, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
(LIST,NOPMAP) or (CLIST,NOPMAP) or (CLIST,PMAP)		Yes	N/A	N/A	N/A
NOBATCH, NOCLIST, NOCOUNT, DMAP, NOLST, PMAP, SOURCE, NOSYMDMP, NOTEST, NOOPT, VERB, XREF(SHORT)	LANGX file	Yes	Suggested for production and test		

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

## Preparing OS/VS COBOL programs

Perform the following steps for compiling your OS/VS COBOL programs:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater, RECFM=VB, BLKSIZE= 1rec1+4 to 32k.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, compile with the NOBATCH, NOCLIST, NOCOUNT, DMAP, NOLST, PMAP, SOURCE, NOSYMDMP, NOTEST, NOOPT, VERB, XREF(SHORT) compiler options. The module is production-ready and can be debugged using Debug Tool for z/OS.
4. Modify the SYSPRINT DD in the compiler step to refer to a file. It can be either a permanent or temporary file. This will be the input to the xxxLANGX utility.
5. Add a step after the compiler step to run the Problem Determination tools xxxLANGX utility. This utility program reads the compiler listing and generates

a LANGX file, which is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the program name of your application program. Equivalent xxxLANGX utilities are available in Debug Tool for z/OS as EQALANGX, in Fault Analyzer for z/OS as IDILANGX and in Application Performance Analyzer for z/OS as CAZLANGX.

6. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.

### **Sample JCL for compiling OS/VS COBOL programs**

A sample job for compiling an OS/VS COBOL program for use with the IBM Problem Determination Tools products is available as member IDISJ005 in data set IDLSIDISAM1.

## **Enterprise PL/I Version 3.7 and later programs**

The following table shows various compiler options that can be used to prepare Enterprise PL/I Version 3.7 and later programs for use with the IBM Problem Determination Tools products (IBM Debug Tool for z/OS, IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for production environments have no significant run-time overhead.

Table 17. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for Enterprise PL/I Version 3.7 and later

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
For Enterprise PL/I Version 3.7: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME, AALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)  For Enterprise PL/I Version 3.8 and later: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME), LISTVIEW(AALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	SYSDEBUG file used by Debug Tool for z/OS and Fault Analyzer for z/OS. LANGX file used by Application Performance Analyzer for z/OS	Although the module is larger than a module compiled with the NOTEST option, you can use the module in production if needed.	Suggested for test. You can also use these options in a production environment if the increased load module size is not an issue.		
AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

### Preparing Enterprise PL/I Version 3.7 and later programs

Perform the following steps for compiling your Enterprise PL/I Version 3.7 and later programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for SYSDEBUG files. This is only needed in test environments where debugging will be performed using LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K).
  2. Allocate one or more LANGX libraries for each environment, such as test and production.
  3. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k.
  4. For all programs, such as batch, CICS, and IMS:
    - In test environments:
      - When using the Enterprise PL/I Version 3.7 compiler:  
For all programs, specify the following compiler options:  
TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME,AALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).
      - When using the Enterprise PL/I Version 3.8 and later compilers:  
For all programs, specify the following compiler options:  
TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME), LISTVIEW(AALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).
- TEST(...) and NOPT are required by Debug Tool.
- The SEPARATE sub-option produces a SYSDEBUG file. Save the SYSDEBUG file created by the compiler for IBM Debug Tool for z/OS and optionally, IBM Fault Analyzer for z/OS.
- The AALL (AFTERALL) sub-option of TEST or LISTVIEW stores program source information in the SYSDEBUG file that contains information after the last preprocessor, such as macros and INCLUDEs. This expanded source information is available in the source window of IBM Debug Tool for z/OS while debugging.
- The other options format the compiler listing as required for the xxxLANGX utility.
- Consider using the TEST(ALL,NOHOOK,SEPARATE) options for best performance and to produce a module that can be debugged. Depending on the policies in your organization, the module can be considered for production.
- In production environments:
    - When using the Enterprise PL/I Version 3.7 or later compiler:  
For all programs, specify NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).
- NOTEST disables Debug Tool, but produces a smaller load module.
- The other options format the compiler listing as required for the xxxLANGX utility to produce a production-ready module that can be used with IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS (but not IBM Debug Tool for z/OS).
5. When a TEST(...SEPARATE) option is used, code a SYSDEBUG DD in the second compiler step as follows:
 

```
//SYSDEBUG DD DSN= sysdebug.pds(pgmname),DISP=SHR
```

This is the source information file for IBM Debug Tool for z/OS, and optionally, IBM Fault Analyzer for z/OS. Save it in the SYSDEBUG library,

and specify a member name that is equal to the primary entry point name or CSECT name of your application program.

6. Modify the SYSPRINT DD in the compiler step. This is the compiler listing. Write the listing to either a permanent or temporary file. This is the input to the xxxLANGX utility.

**Note:** This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by IBM Application Performance Analyzer for z/OS or IBM Fault Analyzer for z/OS. Instead, use the primary entry point name.

7. Add a step after the compile step to run the xxxLANGX utility. This utility reads the compiler listing and generates a LANGX file. This is the source information file for IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS. Equivalent xxxLANGX utilities are available in IBM Debug Tool for z/OS as EQALANGX, in IBM Fault Analyzer for z/OS as IDILANGX and in IBM Application Performance Analyzer for z/OS as CAZLANGX. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name of your application program.
8. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
9. If you compile with the TEST option and will promote these modules into production, promote the SYSDEBUG file for your production environment.
10. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

### **Sample JCL for compiling Enterprise PL/I for z/OS Version 3.7 or later programs**

A sample job for compiling an Enterprise PL/I for z/OS Version 3.7 or later program for use with the IBM Problem Determination Tools products is available as member IDISJ011 in data set IDI.SIDISAM1.

## **Enterprise PL/I Version 3.5 and Version 3.6 programs**

The following table shows various compiler options that can be used to prepare Enterprise PL/I Version 3.5 and Version 3.6 programs for use with the IBM Problem Determination Tools products (IBM Debug Tool for z/OS, IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

Table 18. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for Enterprise PL/I Version 3.5 and Version 3.6

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
<p>Preprocess (1st stage) to expand source, In compile (2nd stage):</p> <p>For Enterprise PL/I Version 3.5: TEST(ALL, SYM, NOHOOK, SEPARATE), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)</p> <p>For Enterprise PL/I Version 3.6: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)</p>	<p>SYSDEBUG file used by Debug Tool for z/OS and Fault Analyzer for z/OS.</p> <p>LANGX file used by Application Performance Analyzer for z/OS</p>	<p>Although the module is larger than a module compiled with the NOTEST option, you can use the module in production if needed.</p>	<p>Suggested for test. You can also use these options in a production environment if the increased load module size is not an issue.</p>		
<p>AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)</p>	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.



## Preparing Enterprise PL/I Version 3.5 and Version 3.6 programs

Perform the following steps for compiling your Enterprise PL/I Version 3.5 and Version 3.6 programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for SYSDEBUG files. This is only needed in test environments where debugging will be performed using LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of 1recl < 32K).
2. Allocate one or more LANGX libraries for each environment, such as test and production.
3. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater,RECFM=VB,BLKSIZE= 1recl+4 to 32k.
4. Run a two-stage compile. The first stage preprocesses the program, so the IBM Problem Determination Tools products have access to fully expanded source code with INCLUDEs and macros. The second stage compiles the program. For all programs, such as batch, CICS, and IMS:

- In the first compile stage, in both test and production environments, specify compiler options MACRO,MDECK,NOCOMPIL,NOSYNTAX,INSOURCE to expand INCLUDEs and macros. The output SYSPUNCH DD will be the input SYSIN DD to the second compile stage.

- In the second compile stage, in test environments,

- When using the Enterprise PL/I Version 3.5 compiler:

For all programs, specify the following compiler options:

TEST(ALL,SYM,NOHOOK,SEPARATE), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).

- When using the Enterprise PL/I Version 3.6 compiler:

For all programs, specify the following compiler options:

TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).

TEST(...) and NOPT are required by Debug Tool.

The SEPARATE sub-option produces a SYSDEBUG file. Save the SYSDEBUG file created by the compiler for Debug Tool (and optionally, Fault Analyzer).

The other options format the compiler listing as required for the xxxLANGX utility.

Consider using TEST(ALL,SYM,NOHOOK,SEPARATE) for best performance and to produce a module that can be debugged. Depending on the policies in your organization, the module can be considered for production.

- In the second compile stage, in production environments, specify compiler options NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).

**Note:** The above options can be used with both the Enterprise PL/I Version 3.5 and Version 3.6 compilers.

NOTEST disables Debug Tool, but produces a smaller load module.

The other options format the compiler listing as required for the xxxLANGX utility to produce a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not Debug Tool for z/OS).

5. When a TEST(...SEPARATE) parm is used, code a SYSDEBUG DD in the second compiler step as follows:



```
//SYSDEBUG DD DSN= sysdebug.pds(pgmname),DISP=SHR
```

This is the source information file for IBM Debug Tool for z/OS, IBM Application Performance Analyzer for z/OS and optionally, IBM Fault Analyzer for z/OS. Save it in the SYSDEBUG library, and specify a member name that is equal to the primary entry point name or CSECT name of your application program.

6. Modify the SYSPRINT DD in the second compiler stage. This is the compiler listing. Write the listing to either a permanent or temporary file. This is the input to the xxxLANGX utility.

**Note:** This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by IBM Application Performance Analyzer for z/OS or IBM Fault Analyzer for z/OS. Instead, use the primary entry point name.

7. Add a step after the compile step to run the xxxLANGX utility. This utility reads the compiler listing and generates a LANGX file. This is the source information file for IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS. Equivalent xxxLANGX utilities are available in IBM Debug Tool for z/OS as EQALANGX, in IBM Fault Analyzer for z/OS as IDILANGX and in IBM Application Performance Analyzer for z/OS as CAZLANGX. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name of your application program.
8. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
9. If you compile with the TEST option and will promote these modules into production, promote the SYSDEBUG file for your production environment.
10. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

### **Sample JCL for compiling Enterprise PL/I Version 3.5 or Version 3.6 programs**

A sample job for compiling an Enterprise PL/I for z/OS Version 3.5 or Version 3.6 program for use with the IBM Problem Determination Tools products is available as member IDISJ006 in data set IDI.SIDISAM1.

### **Enterprise PL/I Version 3.4 and earlier programs**

The following table shows various compiler options that can be used to prepare Enterprise PL/I Version 3.4 and earlier programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the

following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 19. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for Enterprise PL/I Version 3.4 and earlier*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
Preprocess (1st stage) to expand source, In compile (2nd stage): TEST(ALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL))	Expanded source file used by Debug Tool for z/OS, LANGX file used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Suggested for test. (Using Debug Tool in production for this compiler is not recommended.)		
AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL))	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

### Preparing Enterprise PL/I Version 3.4 and earlier programs

Perform the following steps for compiling your Enterprise PL/I Version 3.4 and earlier programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for expanded source files. This is only needed in test environments where debugging will be performed. The library can be any RECFM / LRECL / BLKSIZE supported as input by the compiler.
2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test or production.
3. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater, RECFM=VB, BLKSIZE= 1recl+4 to 32k.
4. Run a 2-stage compile. The first stage preprocesses the program, so the IBM Problem Determination Tools have access to fully expanded source code with INCLUDEs and macros. The second stage compiles the program.
  - In the first compile stage, in both test and production environments:

- Specify compiler options MACRO,MDECK,NOCOMPILE,NOSYNTAX,INSOURCE to expand INCLUDEs and macros.
  - Save the output, the expanded source file, in a permanent file in the expanded source file library and specify *member name = program name*. This is the source information file for Debug Tool for z/OS. The output SYSPUNCH DD will be the input SYSIN DD to the second compiler stage.
  - In the second compile stage, for all programs, such as batch, CICS, and IMS:
    - In test environments, specify compiler options TEST(ALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).  
TEST(ALL) and NOPT are required by Debug Tool. Debug hooks are inserted, which add some runtime overhead. Symbolic data required by Debug Tool is also stored in the module, which can make it significantly larger.  
The other options format the compiler listing as required for the xxxLANGX utility.
    - In production environments, specify compiler options NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)).  
NOTEST disables Debug Tool, but provides the best performance. This produces a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not Debug Tool).  
The other options format the compiler listing as required for the xxxLANGX utility.
  - 5. Modify the SYSPRINT DD in the second compiler stage. This is the compiler listing. Save the compiler listing to either a permanent or temporary file. This will be the input to the xxxLANGX utility.
- Note:** This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by Application Performance Analyzer for z/OS or Fault Analyzer for z/OS. Instead, use the primary entry point name.
6. Add a step after the compiler step to run the xxxLANGX utility. The xxxLANGX utility reads the compiler listing and generates a LANGX file, which is the source information file for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Equivalent xxxLANGX utilities are available in Debug Tool for z/OS as EQALANGX, in Fault Analyzer for z/OS as IDILANGX and in Application Performance Analyzer for z/OS as CAZLANGX. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name or CSECT name of your application program.
  7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
  8. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs

(including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

9. For CICS applications only, if the Debug Tool DTCN transaction will be used to start Debug Tool, link edit the Debug Tool CICS startup exit module EQADCCXT into the application load module to enable Debug Tool in CICS. This is not needed if using the CADP transaction instead of DTCN.

### Sample JCL for compiling Enterprise PL/I for z/OS Version 3.4 or earlier programs

A sample job for compiling an Enterprise PL/I for z/OS Version 3.4 or earlier program for use with the IBM Problem Determination Tools products is available as member IDISJ007 in data set IDL.SIDISAM1.

## PL/I for MVS and VM and OS PL/I programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 4 programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

For the test environment, you need both the listing and the LANGX file (for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). In production, only the LANGX file is suggested.

*Table 20. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for PLI for MVS and VM and OS PLI*

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(ALL), AGGREGATE, ATTRIBUTES (FULL), ESD, LIST, MAP, NEST, NOPT, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	No	Suggested for test. (Using Debug Tool in production for this compiler is not recommended.)	Supported	Supported
	LANGX file	No	N/A	Supported	N/A
NOTEST, AGGREGATE, ATTRIBUTES (FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	Yes	N/A	Supported	Suggested for production and test
	LANGX file	Yes	N/A	Suggested for production and test	N/A

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

### Preparing PL/I for MVS and VM and OS PL/I programs

Perform the following steps for compiling your PL/I for MVS and VM and OS PL/I programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for compiler listing files. This is only needed in test environments where debugging will be performed. Specify LRECL=125 minimum, RECFM=VBA, BLKSIZE= 1recl+4 to 32k.
2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
3. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater, RECFM=VB, BLKSIZE= 1recl+4 to 32k.
4. For all programs, such as batch, CICS, and IMS:
  - In test environments, specify compiler options TEST(ALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL).  
 TEST(ALL) and NOPT are required by Debug Tool. TEST adds debug hooks, which add some runtime overhead. Symbolic data required by Debug Tool is stored in the module, which can make it significantly larger.  
 The other options format the compiler listing as required by Debug Tool and by the xxxLANGX utility.
  - In production environments, specify compiler options NOTEST, AGGREGATE, ATTRIBUTES(FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL).  
 NOTEST disables Debug Tool, but provides the best performance.  
 The other options format the compiler listing as required for the xxxLANGX utility.  
 This produces a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS but not Debug Tool for z/OS.
5. Modify the SYSPRINT DD in the compiler step. This is the compiler listing. Save this to a permanent file. The compiler listing is the input to the xxxLANGX utility and is the source information file for Debug Tool for z/OS
 

**Note:** This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by Application Performance Analyzer for z/OS or Fault Analyzer for z/OS. Instead, use the primary entry point name.
6. Add a step after the compiler step to run the xxxLANGX utility. This utility reads the compiler listing and saves a LANGX file. This is the source information file for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Equivalent xxxLANGX utilities are available in Debug Tool for z/OS as EQALANGX, in Fault Analyzer for z/OS as IDILANGX and in Application Performance Analyzer for z/OS as CAZLANGX. Save it in the LANGX file library and specify a member name that is equal to the primary entry point name of your application program.
7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
8. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger

automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

9. For CICS applications only, if the Debug Tool DTCN transaction will be used to start Debug Tool, link-edit the Debug Tool CICS startup exit module EQADCCXT into the application load module to enable Debug Tool in CICS. This is not needed if using the CADP transaction instead of DTCN.

**Sample JCL for compiling PL/I for MVS and VM programs:** A sample job for compiling a PL/I for MVS and VM program for use with the IBM Problem Determination Tools products is available as member IDISJ008 in data set IDLSIDISAM1.

## z/OS XL C and C++ programs

The following table shows various compiler options that can be used to prepare z/OS XL C and C++ programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 21. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for C++*

Compiler options	Output produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
Preprocess (1st stage) to expand source, In compile (2nd stage): TEST, ATTRIBUTE(FULL), NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF	Expanded source file used by Debug Tool for z/OS, compiler listing used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Suggested for test. (Using Debug Tool in production for this compiler is not recommended.)	Supported	Supported
	Expanded source file used by Debug Tool for z/OS, LANGX file used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Supported	Supported	Supported



Table 21. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for C++ (continued)

Compiler options	Output produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
NOTEST, ATTRIBUTE(FULL), NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF	Compiler listing	Yes	N/A	Suggested for production and test	Suggested for production and test
	LANGX file	Yes	N/A	Supported	Supported
Preprocess (1st stage) to expand source. In compile (2nd stage): DEBUG(FORMAT (DWARF), HOOK(LINE, NOBLOCK, PATH), SYMBOL, FILE(location))	Expanded source file and DWARF file	No	Supported. (Using Debug Tool in production for this compiler is not recommended.)	N/A	N/A

**Notes:**

1. The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.
2. The FORMAT(DWARF) option is supported for z/OS Version 1.6 and higher.

Table 22. Examples of compiler options and source information files supported by IBM Problem Determination Tools products for C

Compiler options	Output produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
Preprocess (1st stage) to expand source, In compile (2nd stage): TEST(ALL), AGGREGATE, NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF	Expanded source file used by Debug Tool for z/OS, compiler listing used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Suggested for test. (Using Debug Tool in production for this compiler is not recommended.)	Supported	Supported
	Expanded source file used by Debug Tool for z/OS, LANGX file used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Supported	Supported	Supported
NOTEST, AGGREGATE, NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF	Compiler listing	Yes	N/A	Suggested for production and test	Suggested for production and test
	LANGX file	Yes	N/A	Supported	Supported
Preprocess (1st stage) to expand source. In compile (2nd stage): DEBUG(FORMAT (DWARF), HOOK(LINE, NOBLOCK, PATH), SYMBOL, FILE(location))	Expanded source file and DWARF file	No	Supported. (Using Debug Tool in production for this compiler is not recommended.)	N/A	N/A

**Notes:**

1. The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.
2. The FORMAT(DWARF) option is supported for z/OS Version 1.6 and higher.

**Preparing z/OS XL C and C++ programs**

Perform the following steps for compiling your z/OS XL C and C++ programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for expanded source files. This is only needed in test



environments where debugging will be performed. This can be any RECFM / LRECL / BLKSIZE supported as input by the compiler.

2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for compiler listing files. Allocate one or more compiler listing libraries for each environment, such as test and production.
3. Create a corresponding listing library for each load library. Specify LRECL=133,RECFM=FBA,BLKSIZE=(multiple of lrecl up to 32k) or LRECL=137 or greater, RECFM=VBA,BLKSIZE= lrecl+4 to 32k.
4. Run a 2-stage compile. The first stage preprocesses the program, so the IBM Problem Determination Tools products have access to fully expanded source code. The second stage compiles the program.
  - In the first compile stage, in both test and production environments:
    - Specify compiler options PP(COMMENTS,NOLINES) to expand INCLUDEs and macros. The output is SYSUT10 DD, which is the expanded source file and is the input for the second compiler stage.
    - Modify the SYSUT10 DD to enable Debug Tool, by saving it in a expanded source library and specify a member name that is equal to the primary entry point name or CSECT name of your application program.
  - For all programs, such as batch, CICS, and IMS, for the second compiler stage:
    - In test environments:
      - For C++, specify compiler options TEST, ATTRIBUTE(FULL), NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF.  
TEST and NOOPT are required by Debug Tool. Debug hooks are inserted, which will adds runtime overhead. Symbolic data required by Debug Tool is stored in the module, which can make it significantly larger.  
The other options format the compiler listing as required by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
      - For C, specify compiler options TEST(ALL), AGGREGATE, NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF.  
TEST(ALL) and NOOPT are required by Debug Tool. Debug hooks are inserted, which adds runtime overhead. Symbolic data required by Debug Tool is stored in the module, which can make it significantly larger.  
The other options format the compiler listing as required by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
    - In production environments:
      - For C++, specify compiler options: NOTEST, ATTRIBUTE(FULL), NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF.
      - For C, specify compiler options: NOTEST, AGGREGATE, NOIPA, LIST, NESTINC(255), NOOFFSET, NOOPT, SOURCE, XREF.  
NOTEST disables Debug Tool, but provides the best performance. This produces a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS, but not Debug Tool for z/OS.  
The other options format the compiler listing as required for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
5. Modify the SYSCPRT DD in the second compiler stage to refer to a file. This is the compiler listing and is the source information file for Fault Analyzer for

z/OS and Application Performance Analyzer for z/OS. Save it in the compiler listing library and specify a member that is equal to the CSECT name of your application program.

```
//SYSCPRT DD DSN=compiler.listing.pds(csect-name),DISP=SHR
```

**Note:** To enable source support in Fault Analyzer, it is a requirement that CSECTs in C programs are named using:

```
#pragma csect(code, "csect_name")
```

where, if using a PDS(E), *csect\_name* matches the compiler listing or LANGX file member name.

6. Modify the promotion process to promote compiler listing files. When a load module is promoted, for example, from test to production, promote the corresponding compiler listing file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the compiler listing file that you perform with the module during promotion.
7. Optionally, include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.
8. For CICS applications only: if the Debug Tool DTCN transaction will be used to start Debug Tool, link edit the Debug Tool CICS startup exit module EQADCCXT into the application load module to enable Debug Tool in CICS. This is not needed if using the CADP transaction instead of DTCN.

## Sample JCL for compiling z/OS C++ programs

A sample job for compiling a z/OS C/C++ program for use with the IBM Problem Determination Tools products is available as member IDISJ009 in data set IDL.SIDISAM1.

## Assembler programs

The following table shows various assembler options that can be used to prepare programs for use with the IBM Problem Determination Tools products (Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate if the load module produced is suitable for a production environment. Load modules suitable for a production environments have no significant run-time overhead.

*Table 23. Examples of assembler options and source information files supported by IBM Problem Determination Tools products for Assembler*

Assembler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for Debug Tool for z/OS	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
ADATA	SYSADATA file	Yes	N/A	Supported	Supported
ADATA	LANGX file	Yes	Suggested for production and test		

**Note:** The highlighted row or rows in the table above indicate the suggested compiler options and source information file types for each product.

## Preparing Assembler programs

Perform the following steps for assembling your programs:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater, RECFM=VB, BLKSIZE= lrecl+4 to 32k.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, specify ADATA.  
ADATA instructs the assembler to produce a SYSADATA file, which contains source and symbolic data about the program. This produces a production-ready module that can be debugged using Debug Tool for z/OS. ADATA does not affect the contents of the assembled module.
4. Add a SYSADATA DD in the assembler step. This file is created by the assembler and it can be a permanent or temporary file. Specify LRECL=8188 or greater, RECFM=VB, BLKSIZE= lrecl+4 to 32k. This file is the input to the xxxLANGX utility.
5. Add a step after the assembler step to run the xxxLANGX utility. The xxxLANGX utility reads the SYSADATA file and creates a LANGX file. The LANGX file is the source information file for Debug Tool for z/OS, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Equivalent xxxLANGX utilities are available in Debug Tool for z/OS as EQALANGX, in Fault Analyzer for z/OS as IDILANGX and in Application Performance Analyzer for z/OS as CAZLANGX.
6. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the CSECT name.
7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
8. If the assembler program is Language Environment-enabled, optionally include a Debug Tool Language Environment exit module into the load module during the linkage editor step. This is one way to enable Debug Tool's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.
9. For CICS programs only: If the program is a CICS main program, is enabled for Language Environment, and the Debug Tool DTCN transaction will be used to start Debug Tool, then supplied module EQADCCXT must be included in the load module during the linkage editor step.

## Sample JCL for assembling a program

A sample job for assembling a program for use with the IBM Problem Determination Tools products is available as member IDISJ010 in data set IDI.SIDISAM1.



---

## 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 311.)

If the relevant file cannot be located as a side file, but a compiler listing is found, then Fault Analyzer will automatically generate a side file from the listing and place it in a temporary data set, which will be 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.

For programs written in High Level Assembler, Fault Analyzer does not use the assembly listing but instead the SYSADATA file 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 will search 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 310.

---

### Creating side files using IDILANGX

To create a side file from a compiler listing, a program named IDILANGX is used.

The sample JCL in Figure 126 on page 304:

- Compiles a COBOL program.

**Note:** You can only compile one program per compile step in order to name the compiler listing PDS(E) member (if using a partitioned data set), and to ensure that only one compiler listing is written to the output file.

## Creating side files using IDILANGX

- Executes IDILANGX to process the listing and store it as a side file where Fault Analyzer can access it.

For return codes issued by IDILANGX, see *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide*.

- Writes the listing as part of the job output.

The sample file is provided as member IDISCMPS in the IDI.SIDISAM1 data set.

```

//IDISCMPS JOB (GSF), 'GENERATE.SIDE.FILE',NOTIFY=&SYSUID.,
//          MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1)
//          JCLLIB ORDER=(IGY.V2R1M0.SIGYPROC) <== INSTALLATION
//*                                               IGYWCLG PROC
//*
//*****
//* THIS JOB RUNS A COBOL COMPILE PLUS PRODUCES A SIDE FILE */
//* FROM A PROGRAM LISTING THAT FAULT ANALYZER CAN USE FOR */
//* OBTAINING Source INFORMATION.                          */
//* THE COMPILE OUTPUT IS THEN WRITTEN TO SYSUT2 IN THE    */
//* IEBGENER STEP.                                         */
//*****
//*
//CBLRUN    EXEC IGYWC,PARM.COBOBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN DD DATA,DLM='##'
...
(Program source not shown)
...
##
//COBOL.SYSPRINT DD DSN=&&COBLIST(IDISCBL1),
//          DISP=(,PASS),SPACE=(TRK,(10,5,5),RLSE),
//          DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//*

```

Figure 126. Sample JCL to compile a COBOL program and store the side file (Part 1 of 2)

```
//IDILANGX EXEC PGM=IDILANGX,REGION=4096K,  
// PARM='IDISCB1 (COBOL ERROR'  
//LISTING DD DISP=(OLD,PASS),DSN=&&COBLIST  
//IDILANGX DD DISP=SHR,DSN=IDI.IDILANGX  
//SYSUDUMP DD SYSOUT=*  
//*  
//IEBGENER EXEC PGM=IEBGENER,REGION=4096K  
//SYSUT1 DD DISP=OLD,DSN=&&COBLIST(IDISCB1)  
//SYSUT2 DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
/*
```

Figure 126. Sample JCL to compile a COBOL program and store the side file (Part 2 of 2)

After you have created and stored a side file, there is no benefit to Fault Analyzer in retaining the listing.

If you already have listings, you can turn them into side files. Here is sample JCL to do this (it is provided as member `IDISFILE` in the `IDI.SIDISAM1` data set):

```
//IDILANGX JOB (C97),'IDILANGX',MSGCLASS=X,
//      CLASS=A,NOTIFY=&SYSUID
//*****
/* This job produces a side file from a program listing that
/* Fault Analyzer can use for obtaining source information.
/* This particular example is set up for a COBOL extraction
/* from IDI.LISTING.COBOL(COBOLA) to IDI.IDILANGX
//*****
//IDILANGX EXEC PGM=IDILANGX,REGION=4096K,
// PARM='COBOLA (COBOL ERROR'
//LISTING DD DISP=SHR,DSN=IDI.LISTING.COBOL 1
//IDILANGX DD DISP=SHR,DSN=IDI.IDILANGX
//SYSUDUMP DD SYSOUT=*
```

Figure 127. Sample JCL to create a side file from a COBOL listing

Notes:

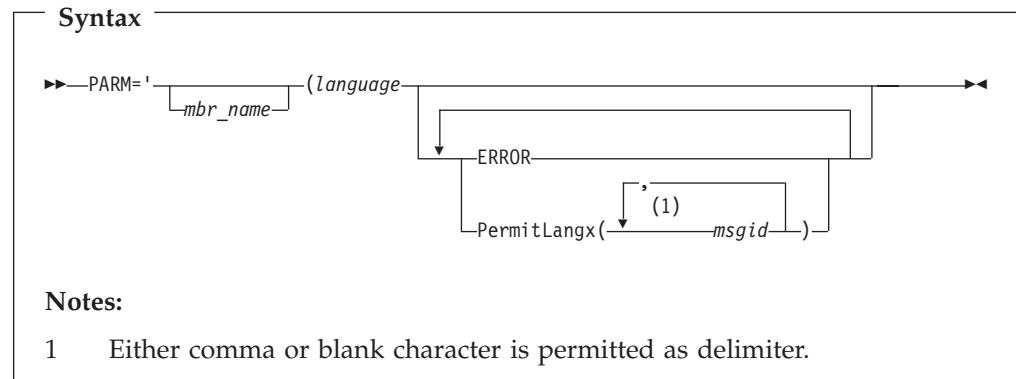
- 1 DDname must be LISTING for all types of compiler listings, or SYSADATA for an assembler SYSADATA file.

For Fault Analyzer processing to work smoothly, the IDILANGX library pointed to in these JCL samples must also appear as a value for the IDILANGX suboption of the DataSets option, or on an IDILANGX DD statement in a reanalysis job.

A compiler listing is the only data format that IDILANGX accepts as input, with the exception of SYSADATA for assembler.

## IDILANGX parameters

The PARM string passed to IDILANGX should contain:



where:

*mbr\_name* is the compiler listing or ADATA file member name in the input data set identified by the LISTING DD name (for a compiler listing) or the SYSADATA DD name (if an ADATA file). This parameter is optional. If omitted, the JCL must specify for the compiler listing or ADATA file, either a sequential data set, or a PDS(E) data set with member name. Also, the output IDILANGX member will be named according to the input program name. In the case of COBOL, for example, this is the name found on the PROGRAM-ID source line.

## Creating side files using IDILANGX

<i>language</i>	is the language of the compiler listing or ADATA file, as: <ul style="list-style-type: none"><li>• COBOL</li><li>• PLI</li><li>• C</li><li>• ASM</li></ul>
<b>ERROR</b>	is an optional parameter that provides additional diagnostics on variables for which information is incomplete.
<b>PermitLangx</b> ( <i>msgid</i> , ...)	is an optional parameter that specifies message IDs for compiler error messages which should be ignored. For details, see “PermitLangx” on page 490.

## Side file compatibility with Debug Tool for z/OS

A side file produced by the Debug Tool EQALANGX<sup>13</sup> program, or the Fault Analyzer IDILANGX program, is usable by both Debug Tool and Fault Analyzer. No special options are required.

**Note:** Debug Tool use of IDILANGX side files is for OS/VS COBOL and Assembler programs only.

---

## Including an IDILANGX step in your SCLM translator

If you use the ISPF/PDF Software Configuration & Library Manager (SCLM) to manage your application software, then you might want to include an IDILANGX step in your SCLM translator, since Fault Analyzer side files generally take up less disk space than compiler listings. Shown in the following, and included in the Fault Analyzer softcopy samples data set, are examples of an IDILANGX step inserted into a High Level Assembler and a COBOL SCLM translator.

### High Level Assembler SCLM example

This example is included in data set IDI.SIDISAM1 as member IDISCLMA.

```
*          SYSADATA DDNAME used in HLASM step.
*          (* SYSADATA *)
          FLMALLOC  IOTYPE=W,DDNAME=SYSADATA,RECFM=VB,RECNUM=9000,    C
              LRECL=8188,BLKSIZE=8192,PRINT=Y
*
*
* IDILANGX BUILD TRANSLATOR
*
          FLMTRNSL  CALLNAM='IDILANGX',                                C
              FUNCTN=BUILD,                                           C
              COMPILE=IDILANGX,                                       C
              DSNAME=IDI.SIDIAUTH,                                    C
              VERSION=3.5.2,                                          C
              GOODRC=0,                                              C
              PORDER=1,                                             C
              OPTIONS='@@FLMMBR(ASM ERROR'
*
*          (* SYSADATA *)
          FLMALLOC  IOTYPE=U,DDNAME=SYSADATA
*
*          (* IDILANGX *)
          FLMALLOC  IOTYPE=P,DDNAME=IDILANGX,DFTTYP=IDILANGX,        C
              KEYREF=OUT2,BLKSIZE=27998,LRECL=1562,RECFM=VB,        C
              RECNUM=10000,DIRBLKS=50,DFTLMEM=*
```

---

13. EQALANGX and IDILANGX are the same program with two different names.



## COBOL SCLM example

This example is included in data set IDI.SIDISAM1 as member IDISCLMC.

```
*****
*      --COPY SYSPRINT FILE TO LISTING
* The COPYFILE EXEC, in dataset PDFTDEV.PROJDEFS.EXEC contains the
* following:
*
* /* REXX */
* /*****
* /* Copy file I to file O. Both are assumed to be pre-allocated. */
* /*****
* PARSE UPPER ARG I", "O .
* "EXECIO * DISKR "I" (STEM R. FINIS "
* "EXECIO * DISKW "O" (STEM R. FINIS "
* RETURN
*
*****
*
      FLMTRNSL  CALLNAM='COPY FILES          ',          C
                FUNCTN=BUILD,                          C
                COMPILE=COPYFILE,                        C
                DSNAME=PDFTDEV.PROJDEFS.EXEC,            C
                CALLMETH=TSOLNK,                         C
                VERSION=1.0,                             C
                PORDER=1,                                C
                OPTIONS=(SYSPRINT,LISTING),              C
                GOODRC=0
*
      FLMALLOC  IOTYPE=W,RECFM=VBA,LRECL=133,            C
                RECNUM=90000,DDNAME=LISTING
*
      FLMTRNSL  CALLNAM='IDILANGX',                      C
                FUNCTN=BUILD,                             C
                COMPILE=IDILANGX,                         C
                DSNAME=IDI.SIDIAUTH,                      C
                VERSION=3.5.2,                            C
                GOODRC=0,                                 C
                PORDER=1,                                 C
                OPTIONS='@@FLMMBR(COBOL ERROR'
*
      (* LISTING *)
      FLMALLOC  IOTYPE=U,DDNAME=LISTING
*
      (* IDILANGX *)
      FLMALLOC  IOTYPE=P,DDNAME=IDILANGX,DFLTYP=IDILANGX, C
                KEYREF=OUT2,BLKSIZE=27998,LRECL=1562,RECFM=VB, C
                RECNUM=10000,DIRBLKS=50,DFTMEM=*
```

---

## 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 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

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++:

ATTRIBUTES (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 will be 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 will usually be 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 can not 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 OFFSET is allowed, in which case Fault Analyzer will not be 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 ATTRIBUTES is a minimum requirement; ATTRIBUTES(FULL) is permitted and has no detrimental effect.
- 5 Although the use of NOLIST and OFFSET is allowed, Fault Analyzer will not be 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.

## TEST option considerations

With all compilers, the additional use of the TEST option may provide program information in addition to what is available via 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 which is suitable for use by Fault Analyzer is written. (See "Locating compiler listings or side files" on page 311 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 an IDILANGX side file created from a compiler listing, then it should be retained for use by Debug Tool for z/OS and Fault Analyzer.

## Required compiler options for IDILANGX

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 will always contain the statement number table (which provides source line offsets), and will also contain 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 may be written to an MVS data set or HFS file. Fault Analyzer will automatically locate 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 CDADBGLD utility is used for this). Like DWARF, MDBG files may 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 Chapter 19, “Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products,” on page 273.

In addition to the information 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 will be 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 461.

---

### 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 will prevent source code information from being determined for these types of programs.

For information about C program CSECT naming requirements, see “z/OS XL C and C++ programs” on page 296.

For information about assembler program CSECT naming requirements, see “Assembler programs” on page 300.

---

### Locating compiler listings or side files

Fault Analyzer will choose 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>14</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.

---

14. See “TEST option considerations” on page 309 for recommended language-dependent suboptions.

## Locating compiler listings or side files

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:
  - a. Compiler Listing Read user exit called with request for Fault Analyzer side file.
  - b. 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:
  - a. Compiler Listing Read user exit called with request for compiler listing (or an Assembler SYSADATA file).
  - b. 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.

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 IDILANGX side file for a program, a number of tests will be 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 will not be used to provide source-level information.

- COBOL-specific tests (excluding OS/VS COBOL)

For COBOL, four additional 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 will be compared to those found in the load module, and if either do not match, then the compiler listing or side file will not be 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 compiled with TEST(NOSEP), 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.

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 Analyzer will prompt the user to optionally supply the location of a compiler listing, Fault Analyzer side file, or COBOL SYSDEBUG side file. For additional details on this prompt, see "Prompting for compiler listing or side file" on page 156.

### 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 324 for an alternative method of activating this trace under CICS.)

The trace information will be written to the IDITRACE DDname destination. An example of compiler listing or side file search trace information follows:

---

```
Listing/Side-file search trace for COBMST5
LE compile date 2002-12-17 time 16-11-10
DA.WDBLANGX
  Rejected - Member not found
  Rejected - Member not found
PATRICK.NIMROD.LISTING.COBOL
  Accepted - Timestamp valid, size test valid
    - Timestamp date 2002-12-17 time 16.11.10
DA.LISTING.COBOL
  - Search not required.
  - Search not required.
```

---

*Figure 128. 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 156.

---

## Compiler listings and side file attributes

Compiler listings and side files must be allocated using the following attributes:

DDname	Attributes
--------	------------



## Compiler listings and side file 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: <ul style="list-style-type: none"><li>• RECFM=VB or VBA and LRECL≥137</li><li>• RECFM=FB or FBA and LRECL=133</li></ul>
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≥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 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 upper-cased 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 58), then CSECT mapping is generally not available. This 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 512). This 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 will use the IDIRLOAD concatenation to search for a module of the same name and use IEWBIND to extract the CSECT name information, thus facilitating potential IDILANGX source mapping for those CSECTs.

It is important that any load libraries allocated to the IDIRLOAD DDname match the load modules being mapped, otherwise incorrect CSECT and source line determination might result.

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.

---

### IDILANGP side file formatting utility

A utility program, IDILANGP, is provided, which can be used to create a readable listing from a Fault Analyzer IDILANGX side file or a SYSDEBUG side file generated by using the COBOL TEST(NONE,SYM,SEPARATE) option. This might be useful if side files, rather than compiler listings, are kept in order to conserve DASD space, as the utility program is able to format the side file in a way that resembles the original compiler listing.

IDILANGP can be executed in two different ways:

- As an ISPF option 3.4 (Data Set List Utility) line command against a sequential Fault Analyzer IDILANGX side file or COBOL SYSDEBUG side file data set, or if the data set is partitioned, as a line command against a member of the data set.

If a sequential COBOL SYSDEBUG side file data set is used, or if the member of a partitioned COBOL SYSDEBUG side file data set does not match any PROGRAM-ID named program contained within it, then a prompt is displayed which permits a program name to be specified.

All Fault Analyzer IDILANGX side files contained in a sequential data set, or a partitioned data set member, are displayed, regardless of whether these match the member name or not.

The output is written to a temporary data set and displayed using ISPF EDIT.

- As a batch job, like the following:

```
//PRTLNGX JOB ...
//STEP1 EXEC PGM=IDILANGP,PARM='parms'
//SYSPRINT DD SYSOUT=*
```

The PARM string passed to IDILANGP should contain;

#### Syntax

```
►►—PARM='data_set_name'—┐
                        └─PROG:program_name—┘►►
```

where:

## IDILANGP side file formatting utility

*data\_set\_name*

is the name of a sequential Fault Analyzer IDILANGX side file or COBOL SYSDEBUG side file data set, or if the data set is partitioned, the data set name with member name included in parentheses.

Examples:

```
MY.SYSDEBUG.SEQ.DS
MY.IDILANGX.PDS.DS(MYPROG)
```

*program\_name*

is the name of a PROGRAM-ID named program contained within a COBOL SYSDEBUG side file. This must be specified if the COBOL SYSDEBUG side file data set is sequential, or if the member name of a partitioned data set does not match the name of any program contained within it.

The formatted listing is written to the SYSPRINT DD. Normal listing file attributes, such as variable-blocked record format and logical record length of 137, are generally appropriate.

## 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 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 will automatically unpack 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 additional step into your compile job. An example of this 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=rexx-exec-library,DISP=SHR
//ISPLLIB DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB DD DISP=SHR,DSN=ISP.SISPSLIB
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//ISPLLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPLLOG DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//ISPLIST DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SYSTSPT 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=PO,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).

The LMCOPY REXX exec, which must reside in the *rex-exec-library* data set as member LMCOPY:

```

/* LMCOPY Rexx */
Address ISPEXEC
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 an IDILANGX side file (see “Creating side files using IDILANGX” on page 303 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 IDILANGX side file using the IDILANGP utility (see “IDILANGP side file formatting utility” on page 315 for details).

## ISPF-packed compiler listings

---

## 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:

- |                |  |
|----------------|--|
| <b>XPCABND</b> | Global user exit using program IDIXCX52 or IDIXCX53. This is the main exit provided to invoke Fault Analyzer for CICS transaction fault analysis.  |
| <b>XDUREQ</b>  | 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.<br><br>The analysis performed by Fault Analyzer at this exit point is the same as for the XPCABND exit point.   |
| <b>LE Exit</b> | LE abnormal termination exit using program IDIXCCEE. This exit is only effective with Language Environment based application programs when the CEEEXTAN exit has been set.<br><br>CICS AKCS abends can be analyzed using this exit, if both of the following are true: <ul style="list-style-type: none"><li>• The failing program is LE enabled</li><li>• An entry exists in the CICS dump table for AKCS, specifying that a transaction dump is to be taken.</li></ul> |

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. This means that, 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 320), or by using the supplied CFA transaction once CICS has initialized (see “Controlling CICS transaction abend analysis” on page 323).

The LE abnormal termination exit, however, requires a modification to LE (see “Configuring Language Environment for CICS to invoke Fault Analyzer” on page 320), 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 will be 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 320.
- \_\_\_ 2. **Define the required programs to your CICS system**  
For details, see “Defining required programs to CICS” on page 320.
- \_\_\_ 3. **Add the required programs to your startup PLT**  
For details, see “Adding the required programs to the startup PLT” on page 320.
- \_\_\_ 4. **Add the required programs to your shutdown PLT**

For details, see “Adding the required programs to the shutdown PLT” on page 321.

### — 5. Define a transaction for Fault Analyzer

For details, see “Enabling dynamic control of analysis of CICS transaction abends” on page 321.

---

## Configuring Language Environment for CICS to invoke Fault Analyzer

Fault Analyzer provides a Language Environment abnormal termination exit for CICS, IDIXCCEE, as an additional method of invoking Fault Analyzer to the CICS XPCABND global exit. This exit is specific to LE U1xxx or U4xxx-type abends—for more information, see “CICS LE abnormal termination CEEEXTAN CSECT exit IDIXCCEE” on page 222. To enable this exit, you must add it to the CEEEXTAN 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 CEEEXTAN
      AND OVERRIDE AS DESIRED  >>>
```

with

```
CEEAXHD      ,User exit header
CEEEXART  TERMEXIT=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 IDIXCEE, is very similar to this exit. If you are going to install both, make sure that you double-check that you have 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 322 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. IDIPLT should be placed at the end of the PLT list, so that Fault Analyzer

## Adding the required programs to the startup PLT

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. Doing this 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 manual, however, you may 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

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.

## Sample definition job

Figure 129 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 may 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)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIPLTD) GROUP(FA) EXECKEY(CICS)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIPLTS) GROUP(FA) EXECKEY(CICS)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIXCX52) GROUP(FA)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIXCX53) GROUP(FA)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIXCCEE) GROUP(FA)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF PROGRAM(IDIXFA) GROUP(FA) EXECKEY(CICS)
  Language(ASSEMBLER) CEDF(NO) DATALOCATION(ANY)
DEF TRANSACTION(CFA) GROUP(FA)
  PROGRAM(IDIXFA) TASKDATALOC(ANY)
DEF MAPSET(IDIXMAP) GROUP(FA)
ADD G(FA) L(list-name)
/*
```

---

Figure 129. 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 IDILED5 USERMOD as described in “Identifying the LE run-time library (++IDILED5)” on page 244.

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 will only perform analysis if invoked via the IDIXCCEE exit.



## Controlling CICS transaction abend analysis

Once the CFA transaction has been 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

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 re-install 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 may optionally pass a command parameter as described in "Using CFA from an MVS console" on page 325. The initial display will be similar to the following:

```

                                Fault Analyzer Control Transaction

Options: I=Install U=Uninstall
                                Current Status/Error Message
_ XPCABND                        Installed
_ XDUREQ                         Installed
_ LE Exit                       Installed
_ SDUMP Screening                Installed

                                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 130. 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

## Controlling CICS transaction abend analysis

accordingly. If there is an active analysis task (as show 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 345.

The PF11 function is described in “IDITRACE under CICS.”

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 131.

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 131. 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 will be used in subsequent FAST duplicate determination. You will be 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 will be 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**:

Display Filter View Print Options Help									
SDSF JOB DATA SET DISPLAY - JOB AS650F1 (JOB31639)							DISPLAY RESET		
COMMAND INPUT ==>							SCROLL ==> CSR		
PREFIX=* DEST=(ALL) OWNER=SIMCOCK SYSNAME=									
NP	DDNAME	StepName	ProcStep	DSID	Owner	C	Dest	Rec-Cnt	Page
	JESMSG LG	JES2		2	SIMCOCK	X		4	
	JESJCL	JES2		3	SIMCOCK	X		259	
	JESYSMSG	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	

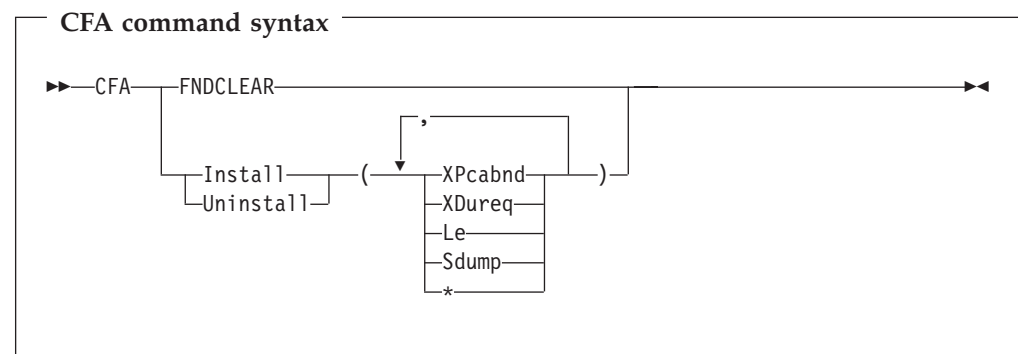
Figure 132. Sample CICS region SDSF Job Data Set display

**Note:** The CFA transaction can only successfully turn IDITRACE off if there is not an IDITRACE DDname in the CICS JCL. This 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 un-allocations.

If the CICS JCL contains an IDITRACE DD statement, then the trace is turned on at the time of starting CICS, and it will remain 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.



### Examples:

- To uninstall all exits from CICS region CICS01, enter the command:

## Controlling CICS transaction abend analysis

```
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 (see description in “Using CFA from a CICS terminal” on page 323), enter the command:

```
/F MYCICS,CFA FNDCLEAR
```

---

## 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 will not call the XPCABND global user exit, and hence Fault Analyzer will not be 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 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. This 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 will attempt 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, and it is invoked using OS linkage convention.

Since IDINDFUE might be invoked from either IDIXCCEE or IDIXCX53, that is, from the CICS XPCABND Global User Exit (GLUE), it must adhere to the CICS rules for coding a XPCABND GLUE program (see the CICS customization guide for details, but significantly, do not use the CICS command-level translator when assembling IDINDFUE).

## CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)

The exit program must be link-edited into a load module named IDINDFUE, and placed into the DFHRPL concatenation of your CICS region if auto load is active, otherwise it must be included in the PPT.

### Entry specifications

Contents of register on entry to IDINDFUE:

Register	Contents
1	Address of input parameter list (see below).
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 24. 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:  <b>0</b> Indicates that the abend should continue to be treated as a duplicate.  <b>1</b> 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 will be 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
```

## CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)

```

        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                Return to Fault Analyzer
        END    IDINDFUE
```

---

## 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 occurs if the IBM-supplied Language Environment default run-time 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 will cause Language Environment to write its diagnostics to the CICS transaction dump data set.

### TERMTHDACT(QUIET)

This will suppress 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 pointed to by the IDIOPTS DDname. For details, refer to “DataSets” on page 458 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 will hold 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. This means 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 will take 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 will not be invoked. This 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 CEEEXTAN CSECT exit IDIXCCEE, which is provided with Fault Analyzer, these types of abends will still be analyzed. See “Configuring Language Environment for CICS to invoke Fault Analyzer” on page 320 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 323 for details about enabling this exit.

---

### Installing the MVS post-dump exit IDIXTSEL

This optional 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 and recovery fault recording SDUMPs.

To install this USERMOD, edit and submit the sample job IDIWTSEL. This will include 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, re-IPL 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 233.

---

### 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 218

---

### Maximizing CICS transaction abend analysis performance

To maximize CICS transaction abend analysis performance, the following should be considered:

## Maximizing CICS transaction abend analysis performance

- 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 234.
- 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 328, if you wish to make it applicable to CICS only.



---

## 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) has been 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 237 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.DBRMLIB.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(DSNTST1) QUALIFIER(DSN8510) MEMBER(DACBD001) -
  ACT(REP) ISOLATION(CS)
RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA51) -
  LIB('DSN510.RUNLIB.LOAD')
RUN  PROGRAM(DACBD001) PLAN(DSNTST1) -
  LIB('CTEST.DB2.LOAD') -
  PARMS('/TERMTHDACT(UADUMP) ')
END
/*
```

---

### DB2 stored procedures

**Note:** Information in this section is based on DB2 version 5. The absence of information for other versions of DB2 does not imply any lack of Fault Analyzer support for these.

If the CEEEXTAN LE abnormal termination exit (IDIXCEE) has not been installed, DB2 stored procedures must be stored with RUNOPTS

## DB2 stored procedures

TERMTHDACT(UATRACE), TERMTHDACT(UADUMP), TERMTHDACT(UAONLY), or TERMTHDACT(UAIMM) to invoke Fault Analyzer via the MVS exit IEAVTABX (IDIXDCAP). For example:

```
INSERT INTO SYSIBM.SYSPROCEDURES
  (RUNOPTS
    )
VALUES ('TERMTHDACT(UADUMP)')
```

When a DB2 stored procedure abnormally terminates, the job executing the calling DB2 program will contain an SQL abend for the DB2 stored procedure in the SYSPRINT output. For example:

```
*** CALL TO storprc NOT SUCCESSFUL:
DSNT408I SQLCODE = -965, ERROR:  STORED PROCEDURE storprc TERMINATED ABNORMALLY
DSNT418I SQLSTATE  = 51021 SQLSTATE RETURN CODE
DSNT415I SQLERRP   = DSNX9CAC SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD   = 0 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD   = X'00000000' X'00000000' X'00000000' X'FFFFFFFF' X'00000000' X'00000000' SQL DIAGNOSTIC
INFORMATION
*** ISSUE ROLLBACK WORK BECAUSE STORED PROCEDURE CALL NOT SUCCESSFUL
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
*** SEVERE ERROR OCCURRED. PROGRAM IS TERMINATING.
```

The generated IDIREPRT after a DB2 stored procedure failure is located within the spooled output of the DB2 SPAS address space. For example:

```
Display Filter View Print Options Help
-----
SDSF JOB DATA SET DISPLAY - JOB DB2SPAS (STC03026) LINE 1-6 (6)
COMMAND INPUT ==> SCROLL ==> CSR
NP DDNAME STEPNAME PROCSTEP DSID OWNER C DEST REC-CNT
JESMSG LG JES2 2 DSN1SPAS S 2
JESJCL JES2 3 DSN1SPAS S 19
JESYSMSG JES2 4 DSN1SPAS S 2
SYSOUT DSN1SPAS 101 DSN1SPAS S 3
CEEDUMP DSN1SPAS 102 DSN1SPAS S 403
IDIREPRT DSN1SPAS 103 DSN1SPAS S 280
```

Fault Analyzer options can be customized via the IDIOPTS DDname within the DB2 SPAS started task. For example:

```
//DSN1SPAS PROC RGN=0K,TME=NOLIMIT,DB2SSN=DSN1,NUMTCB=8
//IEFPROC EXEC PGM=DSNX9STP,REGION=&RGN,TIME=&TME,
// PARM='&DB2SSN,&NUMTCB'
//STEPLIB DD DISP=SHR,DSN=DSN510.RUNLIB.LOAD
// DD DISP=SHR,DSN=DSN510.SDSNLOAD
//IDIOPTS DD DISP=SHR,DSN=DA.DB2SAMP(IDIOPT1)
```

where sample IDIOPT1 contains:

```

Detail(M)
MaxMinidumpPages(128)
DataSets(
    IDILCOB (
        CTEST.DUMPA.LISTING.DB2.COBOL
        CTEST.DUMPA.LISTING.DB2.COB2LE
        CTEST.DUMPA.LISTING.DB2.COB2
    )
    IDILCOBO (CTEST.DUMPA.LISTING.DB2.COBOSVS)
    IDILPLI (CTEST.DUMPA.LISTING.DB2.PLI )
    IDIADATA (CTEST.DUMPA.SYSADATA.DB2 )
)

```

**Note:** The IDIOPTS DD statement is an optional method of providing options to Fault Analyzer. It is not required, but permits options that are specific to the job or started task that contains the DDname to be specified. For more information, see “User options file IDIOPTS” on page 454.

After a stored procedure terminates abnormally, the stored procedure must be restarted via the DB2 subsystem. For example:

```
-START PROCEDURE
```

where '-' is the DB2 subsystem prefix in this example.

See *DB2 for OS/390 Command Reference* for additional information about the DB2 START command.

---

## Improving Fault Analyzer DB2 performance

DB2 does not have an index defined on the SYSIBM.SYSDBRM catalog table. Fault Analyzer will access the SYSIBM.SYSDBRM catalog table whenever it performs analysis for a DB2 fault. To avoid the possibility of poor performance when 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 nnnnnn.DBRMX 1
    ON SYSIBM.SYSDBRM
    (
        PLNAME ASC,
        NAME ASC
    )
    USING STOGROUP mmmmmm 1
        PRIQTY pp 2
        SECQTY ss 2
CLOSE NO;

```

Notes:

- 1** Change the name of the index (*nnnnnn*) and storage group (*mmmmmm*) 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=SYSALLDA,SPACE=(TRK,(1,5))
//SYSLIN   DD DISP=(,PASS),UNIT=SYSALLDA,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
          CEEUOPT 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,FMT=D,TIME=5
//         UNIT=3390,
//         DCB=BLKSIZE=6144
//SYSPRINT DD SYSOUT=*
//DFSIVD1  DD DISP=SHR,DSN=IMS.DFSIVD1
//DFSIVD1I DD DISP=SHR,DSN=IMS.DFSIVD1I
//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 239:

DDname	Data set name
IDIIPJPN	IDI.SIDIPJPN
IDIIMJPN	IDI.SIDIMJPN
IDIISJPN	IDI.SIDISJPN
IDIITJPN	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 will use the ISPF LIBDEF service to logically place the data sets allocated to the IDIIPJPN, IDIIMJPN, IDIISJPN, and IDIITJPN DDnames ahead of the data sets allocated to the ISPLIB, ISPLMLIB, 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 ISPLIB, ISPLMLIB, ISPSLIB, or ISPTLIB is already active at the time of invoking Fault Analyzer, then the existing LIBDEF data sets will be included in the new LIBDEF, after the IDIIPJPN, IDIIMJPN, IDIISJPN, 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, will not be available. (This assumes that only one data set is specified for the IDIIPJPN, IDIIMJPN, IDIISJPN, and IDIITJPN DDname, as will normally be 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 which will be 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 (described on page 479) specifies the national language ID, which is used to select appropriate language-dependant data sets.

To make Japanese the default language for all users, specify the following option in your IDICNF00 parmlib member:

LANGUAGE(JPN)

## Setting the national language



---

## 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 239:

DDname	Data set name
IDIIPKOR	IDI.SIDIPKOR
IDIIMKOR	IDI.SIDIMKOR
IDIISKOR	IDI.SIDISKOR
IDIITKOR	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 will use the ISPF LIBDEF service to logically place the data sets allocated to the IDIIPKOR, IDIIMKOR, IDIISKOR, and IDIITKOR DDnames ahead of the data sets 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 will be 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, will not be available. (This assumes that only one data set is specified for the IDIIPKOR, IDIIMKOR, IDIISKOR, and IDIITKOR DDname, as will normally be 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 which will be 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 (described on page 479) specifies the national language ID, which is used to select appropriate language-dependant data sets.

To make Korean the default language for all users, specify the following option in your IDICNF00 parmlib member:

LANGUAGE(KOR)

## Setting the national language

---

## 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 additional information.

The job assembles and executes a program, which will abend with a system abend code of 0C7.

Since this 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 written to IDIREPRT should contain the following:

---

#### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 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 345 for additional 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 additional information.

The job compiles and executes a COBOL program, which will abend 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 written to IDIREPRT should contain the following:

---

### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 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 IDIVPPLE 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 additional information.

Each job compiles and executes a PL/I program, which will abend and terminate 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 written to IDIREPRT should contain the following:

---

### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 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 IDL.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 IDL.SIDISAM1. Refer to the instructions in the sample job for additional information.

The job compiles and executes a C program in IDL.SIDISAM1(IDISRC1), which will abend and terminate the job step with a return code of 3000.

The synopsis section of the Fault Analyzer report written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 should be in Korean if the Language(KOR) option is in effect.

### End of Korean feature note

A system abend 0C9 occurred in module GO 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 319.

## Verifying the customization of Fault Analyzer under CICS

Having installed the CFA transaction, and the desired exits, invoke the CFA transaction from a CICS terminal, and press PF9 to see the following display:

```

                                Fault Analyzer IVP Testing

Options: S=Select

                                IVP Description
_ 0C1 in program IDIXFA
_ EXEC CICS DUMP DUMPCODE(FAD1) - XDUREQ exit must be installed
_ EXEC CICS ABEND ABCODE(FLT1)
_ EXEC CICS ABEND ABCODE(FLT2) - LE Assembler

Use S to Select the IVP to execute
      PF3=Exit      ENTER=Execute
```

Figure 133. 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:

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 349.

### 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 additional 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 will deliberately abend 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 DSNTEJ8 (JCL) and DSN8OIVP (C source code). This has been modified to deliberately abend while in the connection with DB2 so that Fault Analyzer is invoked and will include 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.

## Verifying the use of Fault Analyzer with DB2

The synopsis section of the Fault Analyzer report written to IDIREPRT should contain the following:

### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 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 IDI.SIDIDOC1 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 additional information.

The job compiles and executes a COBOL program, which will abend 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 has been modified to deliberately abend after having performed DB2 data base access, so that Fault Analyzer is invoked and will include 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 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 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 written to IDIREPRT should contain the following:

#### Japanese feature note

For the Japanese feature of Fault Analyzer, this 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 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:

## Verifying the use of Fault Analyzer with DB2

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:

```
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.DBRMLIB.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 239. This 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 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 wish to view, and press Enter.

For additional information about how to use the Fault Analyzer ISPF interface, refer to Chapter 3, “The Fault Analyzer ISPF interface,” on page 31.

---

## 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 on page 342:

```
//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 which will activate 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), as well as a recovery fault recording fault entry.

Before submitting the job, ensure that the IDIS subsystem has been started.

## Verifying the recovery fault recording set-up

Messages, similar to the following, should be expected:

```
+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.IDIVPC0)
      STORCLAS (SCIDIRFR) MGMTCLAS (PRIMARY) DATACLAS (DEFAULT)
      VOL SER NOS= E$RF01
IGD104I IDIRFRHQ.IDIRFR.FAE1.D081215.T013821.IDIVPC0 RETAINED, DDNAME=SYS00038
IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO IDIRFRHQ.IDIRFR.FAE1.D081215.T013821.IDIVPC0
```

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 (++)IDISRFR)” on page 246.

---

## 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:** Deleting 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) should be avoided as it will cause the history file index to become 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 may 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 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 may 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 may 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.

## Managing history files (IDIUTIL utility)

They follow the same structure as the user exits provided with the Fault Analyzer real time and reanalysis functions. They may 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.

### FILES control statement

#### Syntax

►► FILES—(—<sup>(1)</sup>  
data-set-name—)——►►

#### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating history file data set names.

#### 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 will 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 360.

### LISTHF control statement

#### Syntax

►► LISTHF—(—<sup>& or |</sup>  
ABEND\_DATE—>—yyyy/mm/dd—  
=<—TODAY—days—  
—  
USER\_ID—  
ABEND\_CODE—  
JOB\_NAME—  
CICS\_TRANSACTION\_ID—  
—literal—)——►►



## Description

The LISTHF (LIST History File) control statement is a single keyword 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 may 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
```

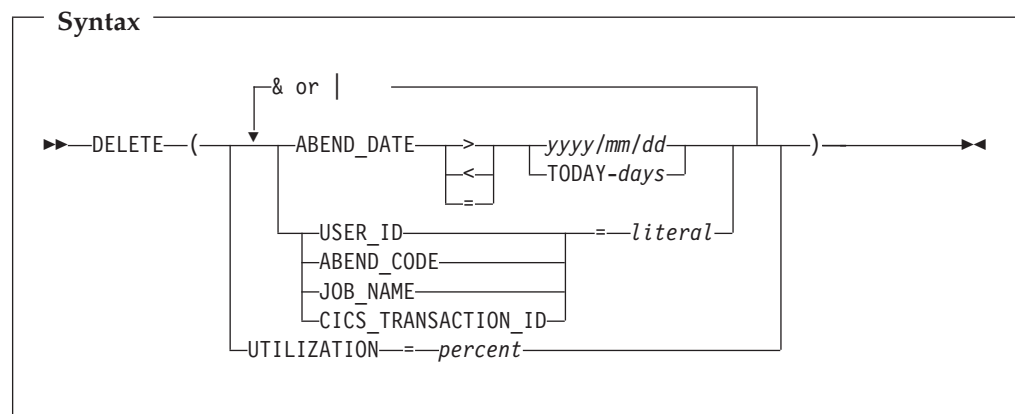
The value specified for *days* must be between 0 and 2147483647, both inclusive.

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 described on page 414 can be used with the LISTHF control statement to apply additional 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 360.

## DELETE control statement



## Description

The DELETE control statement is a single keyword followed by parentheses containing qualifiers to select which fault entries should be deleted.

## IDIUTIL control statements

**Note:** Deletion of a locked fault entry is only possible when overriding the default action using an IDIUTIL Delete user exit. For additional information about the IDIUTIL Delete user exit specification and usage, see “Description” on page 359. 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 78.

The UTILIZATION qualifier can be used to delete entries, starting with the oldest entry, until the specified percentage of utilization is reached. This is only available for PDSE history files. UTILIZATION can not be qualified with additional & or | operators, it must be the only operator in the DELETE statement when used. It may be preceded or followed by any other DELETE statements.

**Note:** When using the UTILIZATION qualifier, all IDIUTIL will do 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. To do this, use the SetMaxFaultEntries(*history-file-name*,*AUTO*) control statement (see “SETMAXFAULTENTRIES control statement” on page 357).

The remaining qualifiers follow the same rules as the LISTHF qualifiers above.

The IDIUTIL Delete user exit described on page 413 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 360 and “Example 3. Deleting history file entries by utilization” on page 360.

## 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 will be changed, followed by up to three alphabetic characters which will become the new prefix characters. Only alphabetic prefix characters are permitted.

Prefixes for all existing fault IDs in the specified history file will be changed to the specified prefix, and all fault IDs later created in this history file will 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 226.

An example showing the use of the SETFAULTPREFIX control statement is provided in “Example 4. Changing history file fault prefix characters” on page 360.

## SETMAXFAULTENTRIES control statement

## Syntax

```

  ►► SETMAXFAULTENTRIES (—data-set-name—, —max-number—) ►►
                                     |
                                     | AUTO
                                     |
                                     |—min-number, THENAUTO—

```

## Description

The SetMaxFaultEntries control statement is a single keyword followed by a set of parentheses containing a PDS(E) history file data set name, whose maximum number of fault entries will be set according to the following:

- If *max-number* is specified, then this is the maximum number of fault entries which can be maintained in the history file. Additional data set extents will be allocated as needed to achieve this number of fault entries. An out-of-space condition will occur 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 *max-number* fault entries.

*max-number* must be a value between 1 and 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 which are not locked will be deleted until the history file only contains the number of fault entries specified.

- If "*min-number*,ThenAUTO" is specified, then the history file will be 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 additional data set extents will generally be allocated and no out-of-space conditions are expected.

This option is available for PDSE history files only—it is not available for PDS history files.

See "AUTO-managed PDSE history files" on page 250 for additional information about AUTO-managed PDSE history files.

*min-number* must be a value between 25 and 2147483647.

- If AUTO is specified, then this is the same as specifying "25,ThenAUTO" (see above).

**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 226.

An example showing the use of the SetMaxFaultEntries control statement is provided in "Example 5. Creating a self-maintained history file" on page 361.

## IMPORT control statement

### Syntax

```

▶▶—IMPORT—(—target-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 target 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 *target* history file. When possible, the imported fault number is left the same, however, the fault number is altered where necessary to ensure no existing target fault entries are overwritten. After successful copy, the fault member is deleted in the *from* data set.

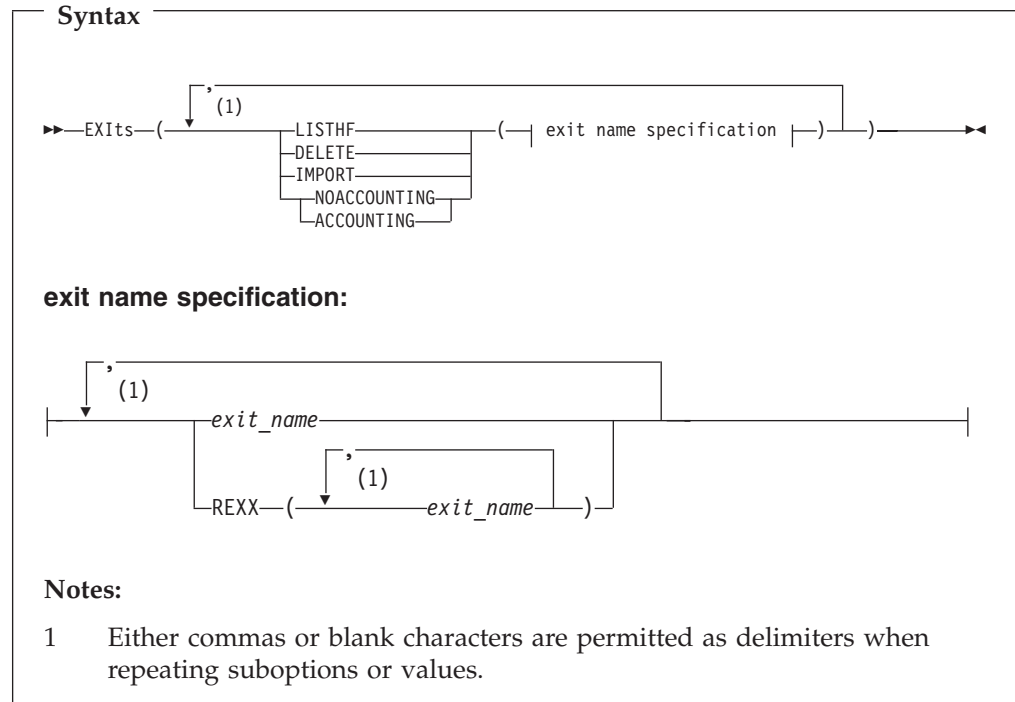
If a single fault ID is included with the *from* data set specification, for example TEMP.HIST(F00234), then only that fault ID will be imported and deleted.

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 253. 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 described on page 412 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 361.

## EXITS control statement



### Description

The Exits control statement follows the same format as the Exits keyword for real-time and reanalysis. The difference is that the exit points are for LISTHF, DELETE, and IMPORT.

The keywords ACCOUNTING and NOACCOUNTING are not exit points, but determine whether or not accounting information is available to the invoked exits through the ENV data area. The default is NOACCOUNTING, which provides a significant performance improvement compared with ACCOUNTING.

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 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 and reanalysis.

A detailed description of each exit type is provided in Chapter 31, "Customizing Fault Analyzer by using user exits," on page 369.

## EXITS control statement

An example showing the use of the Exits control statement is provided in “Example 6. Importing history file entries” on page 361.

---

## 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 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 *
FILES(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 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 *
SetMaxFaultEntries(MY.HIST,100)
/*
```

This facilitates 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 maximum number of faults as per the SetMaxFaultEntries specification.

You might also want to add a SETFAULTPREFIX control statement to your history file creation jobs (see “SETFAULTPREFIX control statement” on page 356) to make fault IDs in each new history file unique.

## 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

## Examples

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'
```

will result 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 additional 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 253.

### 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 412.

In particular, the sample shown in “Example 2” on page 416 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 explanations for application-specific messages

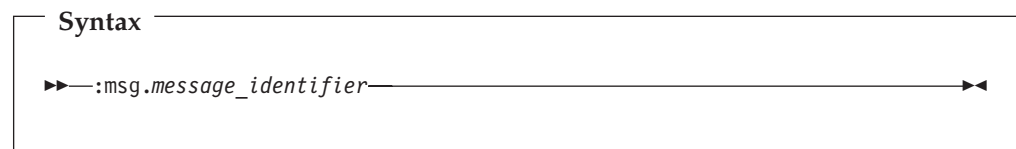
The member IDIHUSRM of the IDI.SIDIDOC1 data set holds user-defined messages.

If one of your applications writes a message identifier to SYSLOG, and the application then abends, Fault Analyzer will look through IDIHUSRM, to see if it can find a message associated with the message identifier. If it finds a match, Fault Analyzer is then able to write out the message, to the "System-Wide Information" section of the analysis report (see "The real-time analysis report" on page 16 for more information).

You can decide what you want to use the messages for. For example, you may use them as check-points when you are developing an application.

The structure of IDIHUSRM is straight-forward. Each message is identified with a header record, followed by the text of the message follows. The text may be in more than one record. The message 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 messages.

The header record has this format:



where

*message\_identifier*

is a code identifying the message. This 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 IDIHUSRM member, then they will automatically be ignored.

Note that ":msg." must begin in column 1.

## Providing explanations for application-specific messages

The following lines of the message are transferred directly as you type them in. So if you include leading spaces in the message, Fault Analyzer will include the leading spaces when it prints the message.

If you begin a message line with `.*`, then Fault Analyzer takes it as a comment and disregards it. It is not included in the analysis report.

In the following example, two messages are defined:

```
:msg.payrollmess1
Processing of accumulated leave about to commence.
- If processing fails after this point,
  check for negative accumulations.
:msg.payrollmess2
Processing of accumulated leave completed.
```

---

## Message search order

There are four sources of messages. When Fault Analyzer is trying to tie a message number to a message and an explanation of the message, it looks in the sources in the following order, stopping if it finds a match:

1. User-defined messages
2. IBM-supplied messages in the IDIDOC data set.
3. IBM-supplied messages in the IDIVSxxx data set, where xxx is the 3-character language ID in effect for the Language option.

Having possibly located the explanation of the message in one of the above sources, the Message and Abend Code Explanation user exit is invoked. This exit may choose to supply an alternate message explanation. For details, see “Message and Abend Code Explanation user exit” on page 386.

---

## Refreshing cached messages

If the Fault Analyzer IDIS subsystem is used, then it is necessary to stop and re-start the IDIS subsystem in order to include new or changed messages in the in-storage cache. If this is not done, then new or changed messages will not be found.

---

## 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 re-apply 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 will 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 is to prevent errors when attempting to load modules from LINKLIST, due to SMP/E having compressed or added extents to the libraries.
2. If Fault Analyzer modules have been placed in LPA using the SETPROG command as opposed to placing IDL.SIDILPA1 in LPA, then do the following:
  - a. Issue the command  
`SETPROG LPA,DELETE,MOD=(IDIDA,IDILANGX),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,IDILANGX),DSN=LNKLST`
3. If IDL.SIDILPA1 is included in your LPALIST, then either re-IPL with the CLPA option or issue the command  
`SETPROG LPA,ADD,MOD=(IDIDA,IDILANGX),DSN=LNKLST`
4. If using CICS, then refresh all installed CICS exits. This can be done by first un-installing, and then reinstalling them, using the CFA transaction as described in “Controlling CICS transaction abend analysis” on page 323.
5. 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 233.
6. Users of the Fault Analyzer ISPF interface should exit and re-enter ISPF for any updates to take effect.

See “Sharing of history files across a sysplex with mixed levels of Fault Analyzer” on page 209 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 will also be prevented from being written.

---

### Temporarily deinstalling Fault Analyzer

If for any reason you wish to temporarily deinstall 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 'I' 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 will not be able to find the expected load modules, and processing will continue 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 can be achieved by adding the following entry to your IFAPRDxx parmlib member:

```
PRODUCT OWNER('IBM CORP')
NAME('FAULT ANALYZER')
ID(5655-W69)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('FAULT ANALYZER')
STATE(DISABLED)
```

## Turning off Fault Analyzer using the IFAPRDxx parmlib member

All parameters except VERSION, RELEASE and MOD must be specified exactly as shown. Any syntactically valid values may 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 re-enabled, 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 may 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  
/*
```

because the JCL switch is processed much earlier by Fault Analyzer than the Exclude option (for details, see “Real-time exclusion processing” on page 25).

---

## 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 re-enable 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 219. 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 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 will affect the current analysis only.

The following user exits are available to users of Fault Analyzer:

- “Analysis Control user exit” on page 377
- “Analysis Control user exit (Dump registration)” on page 380
- “Compiler Listing Read user exit” on page 382
- “Message and Abend Code Explanation user exit” on page 386
- “Formatting user exit” on page 390
- “End Processing user exit” on page 402
- “End Processing user exit (Fault entry refresh)” on page 404
- “Notification user exit” on page 405
- “Notification user exit (Dump registration)” on page 411
- “IDIUTIL Import user exit” on page 412<sup>15</sup>
- “IDIUTIL Delete user exit” on page 413<sup>15</sup>
- “IDIUTIL ListHF user exit” on page 414<sup>15</sup>

Figure 134 on page 370 illustrates the exit points provided for real-time analysis, batch reanalysis, and interactive reanalysis, while Figure 135 on page 371 illustrates the exit points provided for the IDIUTIL batch utility.

---

15. Used with the IDIUTIL batch utility only.

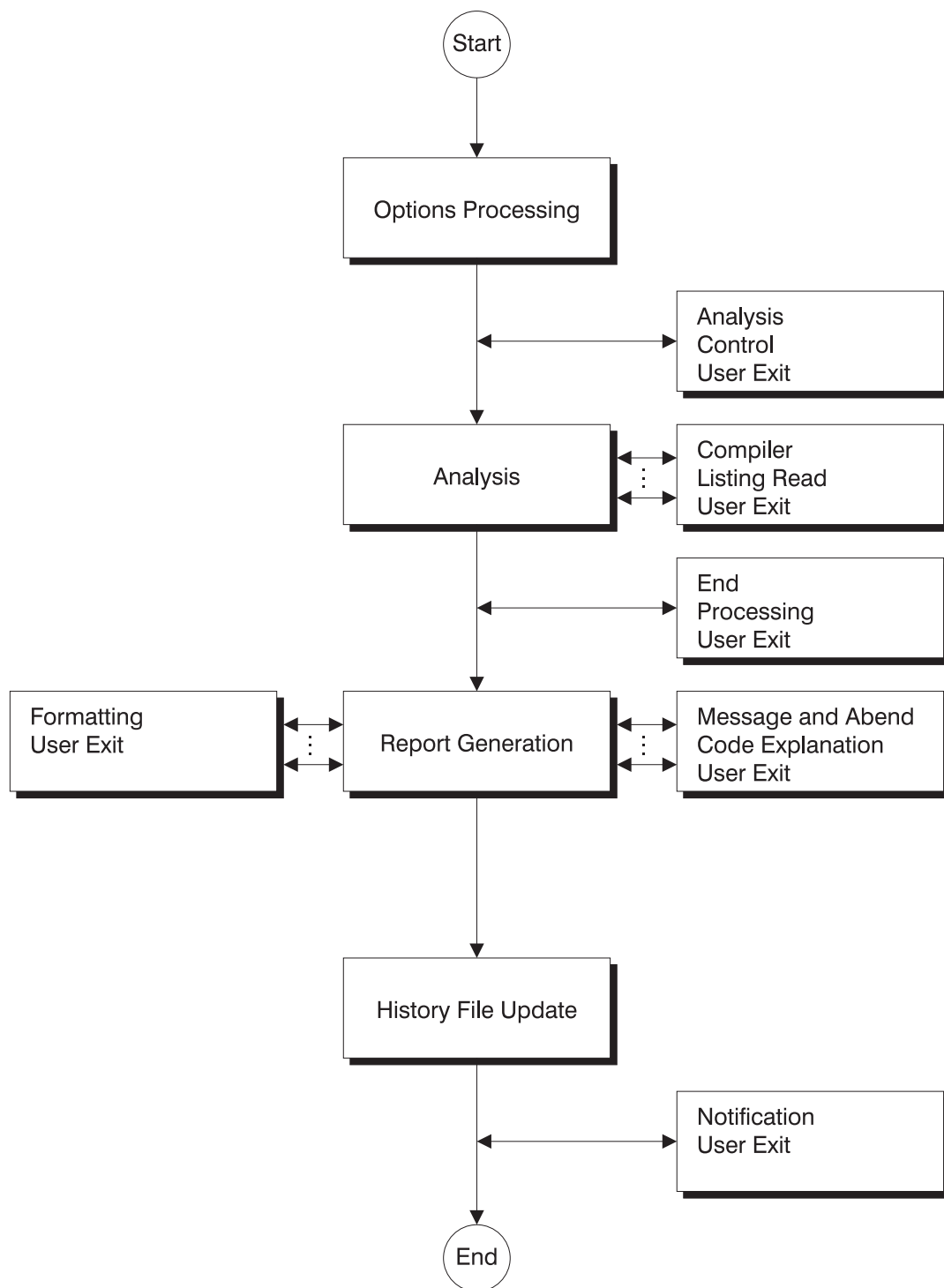


Figure 134. Fault Analyzer analysis exit points (IDIDA)



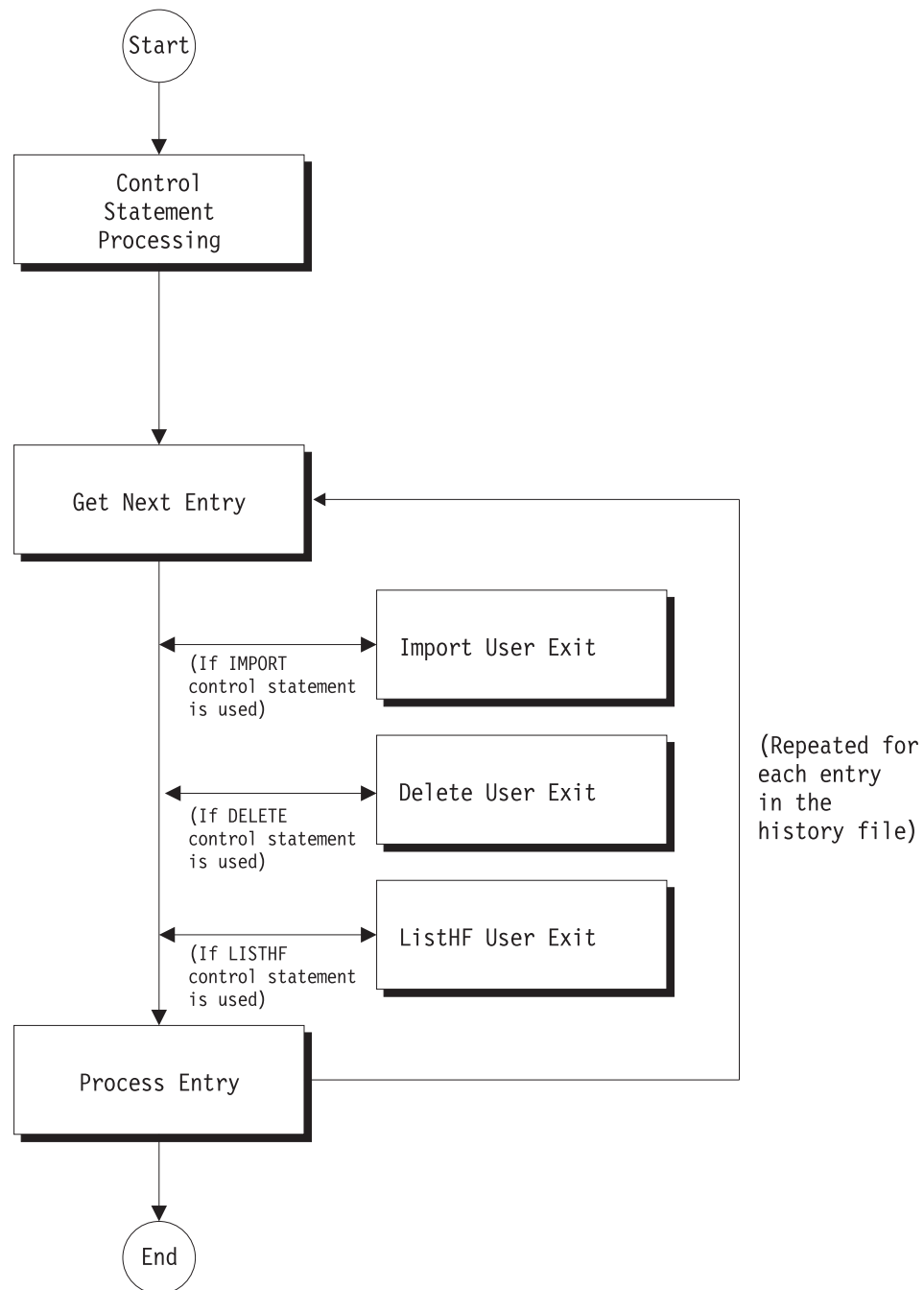


Figure 135. IDIUTIL batch utility exit points

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 493).

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 359.

---

## 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 512.

### 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 377 or “Descriptions of IDIUTIL batch utility user exit types” on page 412.

---

## 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 may be provided in the DataSets option (see page 458), 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 will be 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 373 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 429 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 may not contain an LE main routine (C `main()` function or PL/I `PROC OPTIONS(MAIN)`) as they will be 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
IDISXPLA	Assembler
IDISXPLC	C
IDISXPLB	COBOL

**Note:** Due to COBOL language restrictions, all underscores ('\_') in parameter list field names have been substituted by dashes ('-').

IDISXPLP	PL/I
----------	------

Load module exits may be either reentrant or non-reentrant and should be link-edited `RMODE(ANY)`. They will be 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 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 503.

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 324 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 will be 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 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.

## Diagnostic tracing

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 . . . . . : 0004
EXIT_CALL_TYPE. . . . . : C
FAULT_ID. . . . . :
ABEND_DATE. . . . . : 2001/03/23
ABEND_TIME. . . . . : 10:02:03
DUP_DATE. . . . . :
DUP_TIME. . . . . :
REALTIME. . . . . : Y
SYSTEM_NAME . . . . . : MVSa
JOB_NAME. . . . . : CI03DA
EXEC_PGM_NAME . . . . . : DFHSIP
USER_ID . . . . . : CICSUSER
GROUP_ID. . . . . : APC
ABEND_CODE. . . . . : AEIL
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 . . . . . :
TERMINID. . . . . : SAMA
TCB_ADDRESS . . . . . :
CSA_ADDRESS . . . . . :
TCA_ADDRESS . . . . . :
IDIHIST . . . . . : 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. . . . . : 0000000000
POF_LOADED_FROM . . . . . :
EXEC_LOADED_FROM. . . . . :
MINIDUMP_PAGES. . . . . : 0000000000
CTL.VERSION . . . . . : 0002
Exclude . . . . . :
DETAIL_OPT. . . . . : M
DEFERREDREPORT_OPT. . . . . : N
RETAIN_DUMP_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
IDILCOB0_PRE. . . . . :
```

```

IDILCOBO_JOB. . . . . :
IDILCOBO_CFG. . . . . :
IDILANGX_PRE. . . . . :
IDILANGX_JOB. . . . . :
IDILANGX_CFG. . . . . : FRED.WDBLANGX
IDILPLI_PRE . . . . . :
IDILPLI_JOB . . . . . : USERA.JCLLIB          USERA.TEXT
                                USERA.LOAD
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 . . . . . : 0004
EXIT_CALL_TYPE. . . . . : C
FAULT_ID. . . . . :
ABEND_DATE. . . . . : 2001/03/23
ABEND_TIME. . . . . : 10:02:03
DUP_DATE. . . . . :
DUP_TIME. . . . . :
REALTIME. . . . . : Y
SYSTEM_NAME . . . . . : MVSA
JOB_NAME. . . . . : CI03DA
EXEC_PGM_NAME . . . . . : DFHSIP
USER_ID . . . . . : CICSUSER
GROUP_ID. . . . . : APC
ABEND_CODE. . . . . : AEIL
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
NETNAME . . . . . :
TERMINID. . . . . : 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

```

## Diagnostic tracing

```
RETAIN_DUMP_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.T050
                                FRED.IDILPLIE.T051          FRED.IDILPLIE.T052
                                FRED.IDILPLIE.T053          FRED.IDILPLIE.T054
                                FRED.IDILPLIE.T055          FRED.IDILPLIE.T056
                                FRED.IDILPLIE.T057          FRED.IDILPLIE.T058
                                FRED.IDILPLIE.T059          FRED.IDILPLIE.T060
                                FRED.IDILPLIE.T061          FRED.IDILPLIE.T062
                                FRED.IDILPLIE.T063          FRED.IDILPLIE.T064
                                FRED.IDILPLIE.T065          FRED.IDILPLIE.T066
                                FRED.IDILPLIE.T067          FRED.IDILPLIE.T068
                                FRED.IDILPLIE.T069          FRED.IDILPLIE.T070
                                FRED.IDILPLIE.T071          FRED.IDILPLIE.T072
                                FRED.IDILPLIE.T073          FRED.IDILPLIE.T074
                                FRED.IDILPLIE.T075          FRED.IDILPLIE.T076
                                FRED.IDILPLIE.T077          FRED.IDILPLIE.T078
                                FRED.IDILPLIE.T079          FRED.IDILPLIE.T080
                                FRED.IDILPLIE.T081          FRED.IDILPLIE.T082
                                FRED.IDILPLIE.T083          FRED.IDILPLIE.T084
                                FRED.IDILPLIE.T085          FRED.IDILPLIE.T086
                                FRED.IDILPLIE.T087          FRED.IDILPLIE.T088
                                FRED.IDILPLIE.T089          FRED.IDILPLIE.T090
                                FRED.IDILPLIE.T091          FRED.IDILPLIE.T092
                                FRED.IDILPLIE.T095          FRED.IDILPLIE.T096
                                FRED.IDILPLIE.T097          FRED.IDILPLIE.T098
                                FRED.IDILPLIE.T099          FRED.IDILPLIE.T100
                                FRED.IDILPLIE.T101          FRED.IDILPLIE.T102
                                FRED.IDILPLIE.T103          FRED.IDILPLIE.T104
                                FRED.IDILPLIE.T105          FRED.IDILPLIE.T106
                                FRED.IDILPLIE.T107          FRED.IDILPLIE.T108
                                FRED.IDILPLIE.T109          FRED.IDILPLIE.T110
```

```

FRED.IDILPLIE.T111
FRED.IDILPLIE.T113
FRED.IDILPLIE.T115
FRED.IDILPLIE.T117
FRED.IDILPLIE.T119
FRED.IDILPLIE.T121

IDILPLIE_CFG. . . . . :
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 will be 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 25 shows which exit types that are applicable to each mode of execution.

Table 25. 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)
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

## Analysis Control user exit

- 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 will be 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 will be 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 461), 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 will 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 will 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 will override the current option setting.

The LOCALE option suboption, FADATE, is provided in the CTL.FADATE data area field. Valid changes to this field will 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 will override the current option setting.

Note that a End Processing user exit (see page 402) might later override this option.

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 will 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 additional information on this type of IDIREPRT allocation, see “Combining Fault Analyzer real-time reports” on page 17, “Controlling the SYSOUT class of real-time reports” on page 17, and “Suppressing real-time reports” on page 17.

**When invoked**

This exit is invoked after options processing has completed, and before the commencement of fault analysis.

**Parameters**

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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:

## Analysis Control user exit

- ENV.  
Contains defined symbols for all fields in the ENV data area described on page 512.
- CTL.  
Contains defined symbols for all fields in the CTL data area described on page 505.

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit CTL address in word 2.  
Address of a CTL data area described on page 505.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of an Analysis Control user exit written in REXX.

---

```
/* REXX */
/* Check data areas used */
if ENV.VERSION <> 4 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 136. 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 (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 will be used as the history file for the current fault. If the history file was pre-allocated, it will be 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.

## When invoked

This exit is invoked after options processing has completed, and before the writing of the dump registration fault entry.

## Parameters

See "Parameters" on page 379.

## Example

The following is an example of an Analysis Control dump registration user exit written in REXX.

---

```

/* REXX */
if ENV.VERSION <> 4 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 137. Sample REXX Analysis Control dump registration user exit

## Analysis Control user exit (Dump registration)

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 allocated to the IDIS subsystem<sup>16</sup>, would cause it to be invoked:

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

## 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, etc.). 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 311 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 will be 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 is available in the LST.MODULE\_NAME data area field.<sup>17</sup>
- The load module data set name in the LST.LOAD\_MODULE\_DSN data area field.
- The CSECT name. This is available in the LST.CSECT\_NAME data area field.
- The entry-point name. This 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 will be followed by a tilde (~) and the last character of the untruncated entry-point name.
- The compile date. This is available in the LST.COMPILE\_DATE data area field, in the format YYYY/MM/DD.
- The compile time. This is available in the LST.COMPILE\_TIME data area field, in the format HH:MM:SS.
- The listing type. This 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 fileThe source code information passed back to Fault Analyzer must be of the type specified in this field.
- The language type. This is available in the LST.LANGUAGE\_TYPE data area field, as one of the following:
  - Assembler

---

16. 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 will be ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS region or batch job.

17. See “Parameters” on page 385 for references to the LST data area.

- C/C++
- COBOL
- OS/VS COBOL
- PL/I
- The expected record format of listings or side files provided by the exit. This is available in the LST.RECFM data area field, as V, VB, VBA, F, FB, FBA, etc.
- The expected logical record length of listings or side files provided by the exit. This is available in the LST.LRECL data area field, in decimal character format.

Based on the above information, a user exit may provide the requested listing or side file. This 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 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 will be 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 428.

#### 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.

## Compiler Listing Read user exit

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 428.

An indicator that may 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 will be 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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.  
Contains defined symbols for all fields in the ENV data area described on page 512.
- LST.  
Contains defined symbols for all fields in the LST data area described on page 520.

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit LST address in word 2.  
Address of a LST data area described on page 520.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of a Compiler Listing Read user exit 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 TRSMAN.<sup>18</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.

---

18. TRSMAN is a free IBM compression utility—refer to the web site <http://service.boulder.ibm.com/s390/mvs/tools/packlib/README.HTML> for additional information.

---

```
/* REXX */
if ENV.VERSION <> 4 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
    recfm = strip(LST.RECFM)
    lrecl = strip(LST.LRECL,L,0)
    "IDIALLOC DD(OUTFILE) DSN(myid.TEMP) NEW CATALOG SPACE(1,1) ",
      "RECFM("||recfm||") LRECL("||lrecl||") UNIT(SYSALLDA)"
    address linkmvs "trsmain unpack"
    address mvs "execio * disk 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 138. 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, will receive 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>19</sup> will be 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.

---

19. See "Parameters" on page 389 for references to the XPL data area.



This 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 will be 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 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 is available in the ABEND\_REASON\_CODE field as an 8-character hexadecimal value.
- The abend module name.  
This is available in the ABEND\_MODULE\_NAME field.
- The abend type.  
This is available in the ABEND\_TYPE field which will contain one of the following values:
  - C**     Indicating a CICS transaction abend.  
The field ABEND\_CODE will contain a 4-character CICS abend code.
  - D**     Indicating a CICS dump code.  
The field ABEND\_CODE will contain a 4-character CICS dump code.
  - S**     Indicating a system abend.  
The field ABEND\_CODE will contain a 3-character hexadecimal left-justified system abend code.
  - U**     Indicating a user abend.  
The field ABEND\_CODE will contain a 4-character decimal user abend code.

Based on the above information, a user exit may provide a missing explanation or a replacement of the one found by Fault Analyzer. This 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 428.

### 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) ;
...
```

```
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 428.

The total number of characters that may form the message or abend code explanation may 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* will result 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 will be performed by Fault Analyzer. All records will be presented in the analysis report exactly as they were provided by the Message and Abend Code Explanation user exit.

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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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 described on page 512.
- XPL.  
Contains defined symbols for all fields in the XPL data area described 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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit XPL address in word 2.  
Address of an XPL data area described on page 530.

## Message and Abend Code Explanation user exit

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of a Message and Abend Code Explanation user exit written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 4 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 139. 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:** REXX is the only supported programming language for exits of this type.

### 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.

There are two different ways to invoke exits of this type:

1. By using the Exits option.

This 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.

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 will be used.

2. By using the EXEC command.

This is only available from the interactive reanalysis report. For details on using the EXEC command, see "EXEC" on page 63.

The Formatting exit is initially provided with information about the point-of-failure event in the exit-specific UFM data area.<sup>20</sup> However, information for any event can be obtained by using the IDIEventInfo command. This is described in "IDIEventInfo command" on page 425.

Information about the existence of a load module, including its load address and length, can be obtained using the IDIModQry command. This is described in "IDIModQry command" on page 426.

To obtain storage from the analyzed environment, or to write data to the report, additional 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 418.

**List** This command is used to print storage in the report.

Details of this command are provided in "List command" on page 430.

**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 432.

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 433.

The tagged text is passed back to Fault Analyzer using the IDIWrite command. This can be done in three different ways:

1. Using a quoted string

Example:

```
IDIWrite '<p>Paragraph text.'
```

Either single quotes (') or double quotes (") may 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'
```

---

20. See "Parameters" on page 392 for references to the UFM data area.

## Formatting user exit

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(UFM.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 154.

## Parameters

Fault Analyzer will initialize 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 described on page 512.

- UFM.

Contains defined symbols for all fields in the UFM data area described on page 523.

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.

## Example 1

The following is an example of a Formatting user exit 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 <> 4 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 TCBTIME 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

/*-----*/

```

## Formatting user exit

```

/* ProcessSiblings: Check if the TCB has and siblings and if so process */
/*      those siblings plus and 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 TCBTIME and convert to seconds. */
/*-----*/
GetTime:Procedure
Arg TCBAAddress
TCBTIME = d2x(x2d(TCBAAddress)+x2d('13C'))
"EVALUATE address("TCBTIME") 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 TCBAAddress
FoundLastRB = 0
RB = d2x(x2d(TCBAAddress)+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 = TCBAAddress Then
    FoundLastRB = 1
  Else
    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
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')"
```

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 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 shoe 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 <> 4 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!'
```

## Formatting user exit

```

"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))"
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)

The following is a sample Formatting user exit written in REXX, specifically intended for 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(THLEEN)"
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>' "
```

## Formatting user exit

```
        End
      End
    Otherwise
      Do
        PEMAddress = PA.1
        PEMLength = PL.1
        Call Discover_Hogan_Information
      End
    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
  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
  End
Return(Version)

/*-----*/
/* Initialise: */
/*-----*/
Initialise:

PEMModule.0 = 7
```

```

PEModule.1 = 'BATCHPEM'
PEModule.2 = 'CICSPEM'
PEModule.3 = 'BMPPEM'
PEModule.4 = 'DLIPEM'
PEModule.5 = 'IMSPPEM'
PEModule.6 = 'IMSPPEMF'
PEModule.7 = 'SYSB2PEM'

PEMod = 0
PEAddress = 0
PELength = 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 = 'TS0'
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

```

## Formatting user exit

```

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))"
      TWAAAddress = x
      If TWAAAddress > 0 Then
        Do
          "EVALUATE address("TWAAAddress")",
            "length("TWALength")",
            "REXX(STORAGE(X))"
          ITCBAddr = Substr(x,1,8)
          "IDIWRITE '<P>ITCB Address <addr " ITCBAddr,
            "></addr></P>'"
        End
      End
    End
  Else
    Do
      "IDIWRITE '<P>User TCA address must be provided for CICS SDUMP.</P>'"
    End
  End
End
/* Not CICS system or transaction */
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))"
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 be 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 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(HOGAN)))

```

Alternatively, the exit could be invoked from the interactive reanalysis report by entering the EXEC HOGAN command.

### Example 4 (Batch MVS dump registration)

The following is a sample Formatting user exit written in REXX, which can be used with a MVS dump analysis batch job to create a fault entry in a designated history file. Optionally, the exit can write a WTO message and/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@aul.ibm.com' /* <=== change */
histdsn = 'idi.hist' /* <=== change */
smtpdest = 'PRDSYS0.SMTP' /* <=== change */
smtpdomain='aul.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' ENV.FAULT_ID,
        'created in history file' ENV.IDIHIST,
        'for job' ENV.JOB_NAME,
        'abend' ENV.ABEND_CODE,
        'which occurred on' ENV.ABEND_DATE,
        'at' ENV.ABEND_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

```

## Formatting user exit

The following is a sample job which 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>)'
//IDIEEXEC DD DISP=SHR,DSN=IDI.SIDIEEXEC
//IDIOPTS DD *
//          Exits(Format(REXX(IDISUFM5)))
/*
```

## 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 selected, the End Processing user exit is invoked repeatedly until the history file name is no longer changed. On each re-invocation, duplicate fault information will be updated for the history file specified on the previous invocation.

If an invalid history file is provided in the ENV.IDIHIST field, then no re-invocation of the End Processing user exit will occur, and the history file that was current when the End Processing user exit was last invoked will be 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 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 will be initialized to the number of minutes elapsed since recording of the last duplicate fault. The value will be in the range from 0 to 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 will show the total number of times that a fault was deemed a duplicate of the recorded fault, not including the current fault.
- The field ENV.FAULT\_ID will identify the recorded duplicate fault by its fault ID.

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 will be 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 will be incremented by one and message IDI0044I will be issued.



**Note:** CICS or IMS fast duplicate fault suppression can not be controlled using this exit. Only the NoDup(CICSFAST(...)) or the NoDup(ImageFast(...)) option (see 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 will still be 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 will suppress 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 may be overridden by the End Processing user exit:

- The EPC.SUPPRESS\_MINIDUMP field. If set to 'Y', no minidump will be written to the history file. If set to 'N', the minidump will be written regardless of its size.
- The EPC.SUPPRESS\_FAULT\_ENTRY field. If set to 'Y', no recording of the current fault will be made in the history file (including the minidump). If set to 'N', fault recording will be performed and the minidump may be written subject to the EPC.SUPPRESS\_MINIDUMP field.

- **Dump suppression.**

The End Processing user exit may 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 13 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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 described on page 512.
- EPC.  
Contains defined symbols for all fields in the EPC data area described on page 519.

## End Processing user exit

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit EPC address in word 2.  
Address of a EPC data area described on page 519.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of a End Processing user exit written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 4 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 140. 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 will suppress 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 may be overridden by the End Processing user exit. If set to 'Y', no minidump will be written to the history file. If set to 'N', the minidump will be written regardless of its size.

## End Processing user exit (Fault entry refresh)

The End Processing user exit may 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 will be performed (including the minidump). If set to 'N', the refresh will be performed and the minidump may 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 403.

### Example

The following is an example of a fault entry refresh End Processing user exit written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 4 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 141. 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>21</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 ENV.FAULT\_ID data area field.

Fault Analyzer issues message IDI0003I to indicate the assigned fault ID and history file.

---

21. See “Parameters” on page 406 for references to the NFY data area.

## Notification user exit

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 additional details regarding this field.

### R Recovery fault recording

This 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 28.)

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 ENV.FAULT\_ID data area field.

Fault Analyzer issues message IDI0126I to indicate the assigned fault ID and history file.

### N Normal duplicate.

This 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 ENV.FAULT\_ID data area fields.

The DUPCOUNT field will be set to 1.

### F Fast duplicate (CICS).

This 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 ENV.FAULT\_ID data area fields.

The DUPCOUNT field will be 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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 described on page 512.

- NFY.

Contains defined symbols for all fields in the NFY data area described 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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit NFY address in word 2.  
Address of a NFY data area described on page 522.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

**Calling a non-REXX logging routine from REXX:** For a Notification user exit written in REXX that, for example, calls an external logging routine that is not written in REXX, two additional stem variables are provided which can be used to more easily 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 (for details, see "Load module exits" on page 372). 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 written in REXX which can be used to issue a TSO SEND message.

---

```
/* REXX */
if ENV.VERSION <> 4 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" ENV.FAULT_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 142. 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)))
```

### Example 2

The following is an example of a Notification user exit written in REXX to send an e-mail message via SMTP.

---

```

/* REXX */
if ENV.VERSION <> 4 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 "helo 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 "ENV.FAULT_ID" assigned in history file"
queue strip(ENV.IDIHIST)" for job "ENV.JOB_NAME
queue "program "ENV.EXEC_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 143. 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 written in REXX to extract and show all lines of the captured real-time synopsis.

---

```
/* REXX */
if ENV.VERSION <> 4 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 144. 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 written in REXX which can be used to submit a batch reanalysis job to generate a saved report for a DeferredReport fault entry.

---

```
/* REXX */
if ENV.VERSION <> 4 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 145. 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 (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 ENV.FAULT\_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 SVC dump fault entry in the history file.

### Parameters

See "Parameters" on page 406.

### Example

The following is an example of a dump registration Notification user exit written in REXX.

---

```

/* REXX */
if ENV.VERSION <> 4 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" ENV.FAULT_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 146. 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.

## Notification user exit (Dump registration)

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 is done by setting the UTL.PERFORM\_ACTION data area field<sup>22</sup> to 'Y' if the entry should be imported, or to 'N' if not. By default, the field UTL.PERFORM\_ACTION will always be 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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 described on page 512.
- UTL.  
Contains defined symbols for all fields in the UTL data area described on page 529.

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit UTL address in word 2.

---

<sup>22</sup>. See "Parameters" for references to the UTL data area.

Address of a UTL data area described on page 529.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of a IDIUTIL Import user exit written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 4 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 147. 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 355). This is done by setting the data area field UTL.PERFORM\_ACTION<sup>23</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 which 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

---

23. See “Parameters” for references to the UTL data area.

## IDIUTIL Delete user exit

Fault Analyzer will initialize 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 described on page 512.
- UTL.  
Contains defined symbols for all fields in the UTL data area described on page 529.

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit UTL address in word 2.  
Address of a UTL data area described on page 529.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example

The following is an example of a IDIUTIL Delete user exit written in REXX.

---

```
/* REXX */
if ENV.VERSION <> 4 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 148. 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 354. This is done by setting the data area field UTL.PERFORM\_ACTION<sup>24</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 which 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

The way in which parameters are passed to the exit depends on the exit type—REXX or load module.

Fault Analyzer will initialize 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 described on page 512.
- UTL.  
Contains defined symbols for all fields in the UTL data area described on page 529.

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 will contain the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.  
Address of an ENV data area described on page 512.
- 31-bit UTL address in word 2.  
Address of a UTL data area described on page 529.

**Note:** The high-order bit will be on to indicate that this is the last parameter passed.

### Example 1

The following is an example of a IDIUTIL ListHF user exit written in REXX.

---

24. See “Parameters” for references to the UTL data area.

---

```
/* REXX */
if ENV.VERSION <> 4 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 149. 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

```
//IDIEXEC 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 a IDIUTIL ListHF user exit 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 written by the provided sample will include the columns:

- Fault ID
- Date
- Time
- Lock
- Username
- User Title
- CPU Sec

See “Available columns” on page 42 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 ENV.CPU\_HSECONDS on page 516 for additional 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.

---

```

/* First ensure that the current data area versions match the */
/* versions as at the time of coding the exit. */
If ENV.VERSION <> 4 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(ENV.FAULT_ID))||ENV.FAULT_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(ENV.ABEND_DATE,',',10)
abend_time=INSERT(ENV.ABEND_TIME,',',8)
lock_flag =INSERT(ENV.LOCK_FLAG,',',4)
user_name =INSERT(ENV.USER_NAME,',',8)
user_title=INSERT(ENV.USER_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 150. 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=*
//MYREP DD SYSOUT=*
//COMMA DD SYSOUT=*
//IDITRACE DD SYSOUT=* (Optional)
//IDIEXEC DD DISP=SHR,DSN=IDI.SIDISAM1
//SYSIN DD *

```

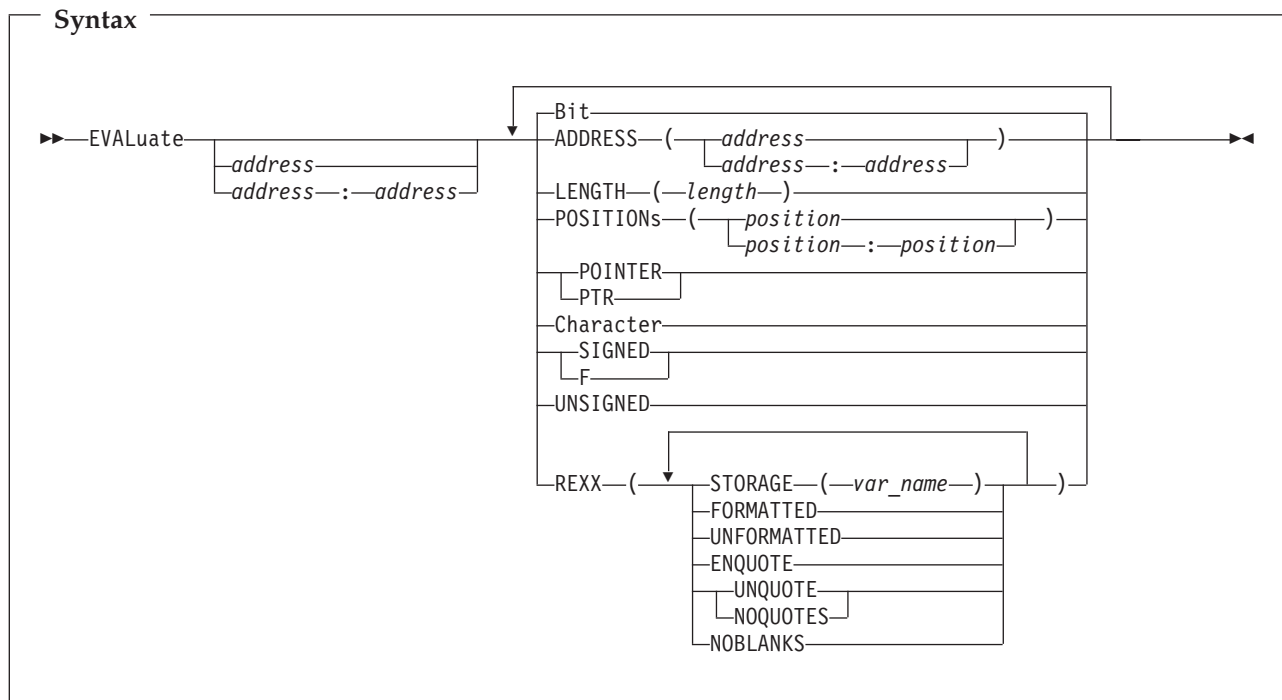
```
Exits(LISTHF(REXX(IDISUTL1)))
FILES(my.histfile)
LISTHF
/*
```

## 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.



### Parameters

*address*

*start:end*

Specifies either the start address, or an address range, as a positional parameter. This 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 returned. This can be specified as either a single address or as an address range.

All addresses must be 1 to 8 hexadecimal digits, optionally followed by a period (for example, "000176C0.").



**LENGTH(*length*)**

Specifies the number of bytes to be returned. This can be specified 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. This can be specified 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 from 1 to 4 bytes. If LENGTH is greater than 4, then it will be 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 will be changed to 2. If LENGTH is greater than 4, then it will be 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 from 1 to 4 bytes. If LENGTH is greater than 4, then it will be 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 is the default.

## Evaluate command

### 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

---

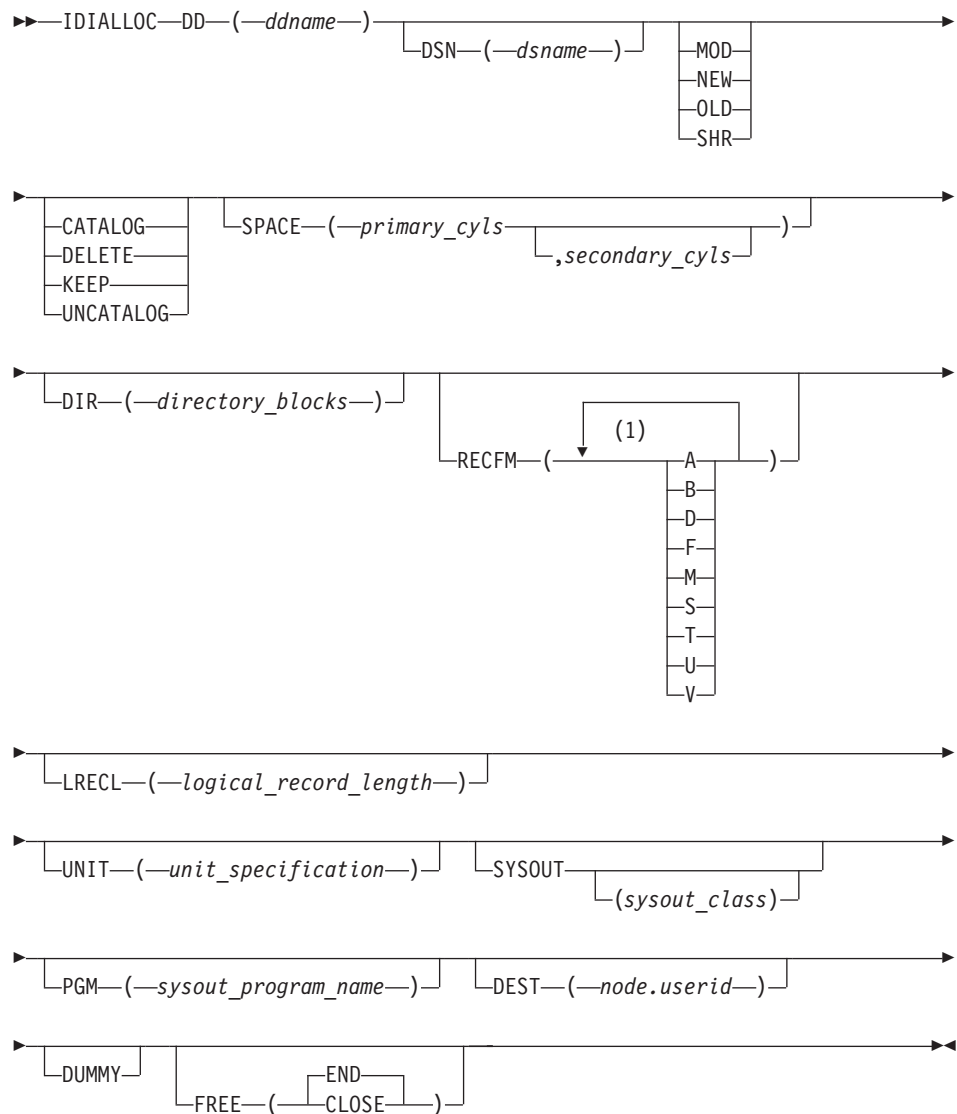
```
/* REXX */  
  
"EVALUATE 0 LENGTH(128) REXX(STORAGE(x))"  
if RC = 0 then say 'Storage at address 0 =' x
```

---

*Figure 151. Evaluate command example*

## IDIALLOC command

The IDIALLOC command can be used to perform dynamic allocation of data sets to DDnames.

**Syntax****Notes:**

- 1 No commas or blank characters may delimit repeated values. For example, to specify fixed-blocked record format, use RECFM(FB) - not RECFM(F B).

**Operands:**

- DD** DDname to be associated with data set. This 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.

## IDIALLOC command

**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 may be used for local destinations.

### DUMMY

Allocate dummy data set.

**FREE** Unallocation specification:

**CLOSE** Requests that the system unallocate the data set when it is closed.

**END** Requests that the system unallocate the data set at the end of the last step that references the data set. This 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.
- 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 may not be opened for read. If the member is opened for read, a system abend S013 will occur. |
| 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 152. 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 153. 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 458) 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 which will replace the current IDIDSECT concatenation. Multiple data set names must be separated by one or more blanks.

## 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

```

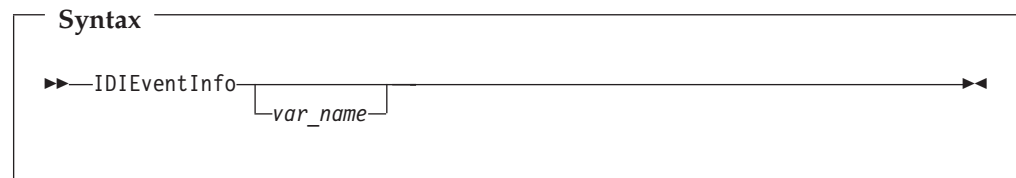
---

Figure 154. 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, etc., 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.



## 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 155. IDIEventInfo command example

## IDIFREE command

The IDIFREE command can be used to unallocate (free) DDnames that might have been allocated using IDIALLOC.

### Syntax

The diagram shows the syntax for the IDIFREE command. It starts with 'IDIFREE' followed by 'DD' and an opening parenthesis '('. Inside the parentheses is 'ddname'. A box labeled '(1)' with an arrow points to the space between 'DD' and '(', indicating that either commas or blank characters are permitted as delimiters. The command ends with a closing parenthesis ')'. The entire command is flanked by double arrows pointing outwards.

### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

### 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 156. IDIFREE command example

## 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*)

is the name of the module for which the load point, and optionally module length, are being requested.

**LP**(*var\_name*)

is the name of a REXX variable to hold the character format of the module load point.

**MODLEN**(*var\_name*)

is the name of a REXX variable to hold the character format of the hexadecimal module length. This 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 157. 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 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.

**Notes:**

1. This command should not be used to register fault entries in a dump registration Analysis Control or Notification user exit, since this could result in additional unwanted fault entries being created. The IDIXTSEL dump registration processing automatically creates a fault entry, even if no user exit is used.

## IDIRegisterFaultEntry command

- As an alternative to using the IDIRegisterFaultEntry 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 476.

### Syntax

```
►► IDIRegisterFaultEntry 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 will be issued to allow the specification of the history file to be used.

For batch reanalysis, this parameter must be specified; otherwise, RC=4 will be issued.

### Return codes

The IDIRegisterFaultEntry 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'
"IDIRegisterFaultEntry IDIHIST("dsn")"
if rc <> 0 then
  "IDIWTO IDIRegisterFaultEntry failed, rc="rc
exit 0
```

Figure 158. IDIRegisterFaultEntry command example

See “Example 4 (Batch MVS dump registration)” on page 401 for an additional example of the IDIRegisterFaultEntry 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*

is 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 one listing record to Fault Analyzer */
rec = 'This is a listing record'
LST.DATA_LENGTH = length(rec)
LST.DATA_BUFFER = rec
"IDIWRITE"
if RC = 0 then say 'Success!'

```

---

Figure 159. 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 will be 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
```

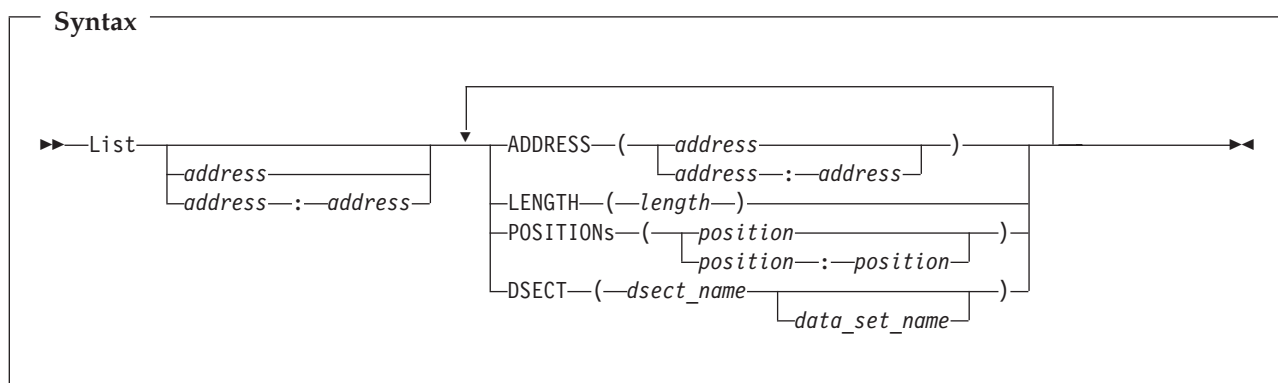
---

Figure 160. IDIWTO command example

To avoid possible confusion, it is recommended that any WTO messages issued do not include a message ID that could be mistaken for one of the formal messages issued by Fault Analyzer. If you wish to prefix message IDs with “IDI”, then you might add an additional 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 to 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 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. This can be specified as either a single address or as an address range.

All addresses must be 1 to 8 hexadecimal digits, optionally followed by a period (for example, "000176C0.").

**LENGTH(*length*)**

Specifies the number of bytes to be printed. This can be specified 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. This can be specified 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 458). 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 145.

**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)"
```

---

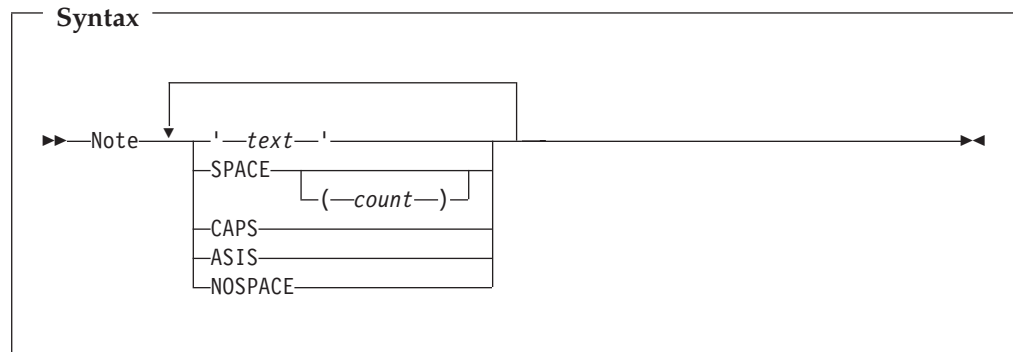
*Figure 161. List command example*

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 by a Formatting user exit to print a line of text.



### 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 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 162. 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, etc. 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:

---

```

"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 163. 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
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 164. 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 will implicitly cause a <P> tag to be inserted ahead of the text.

## Formatting tags

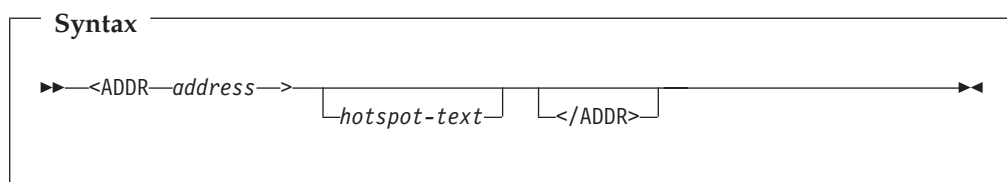
- All tags and attributes are non-case-sensitive.
- The maximum line width of any output is 132 characters. Beyond this, the text will wrap.

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.



#### address

This is the hexadecimal address to be displayed when selecting the point-and-shoot field.

A 64-bit address may be specified using an underscore between bits 31 and 32. For example, 1\_00100000. Note that leading zeroes are implied on either side of the underscore to make each address component 32 bits.

#### hotspot-text

This is the point-and-shoot field text to be displayed. If not specified, then the address is used.

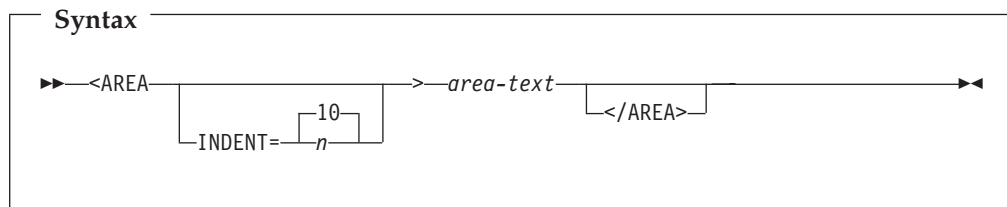
### Description

The ADDR tag does not cause a line break.

In the interactive reanalysis report, the point-and-shoot field text will be 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.



#### INDENT

This attribute specifies the number of characters by which the current indentation will be incremented.

**area-text**

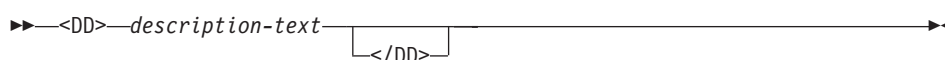
These are tags and text for the display section.

**Description**

The AREA tag can be nested within another AREA tag. Any other tags will be 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 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 is generally only used if the text to be written contains characters that might otherwise be misinterpreted as formatting tags.

**Syntax**


```
>><DATAlength><<
```

**length** This is 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**

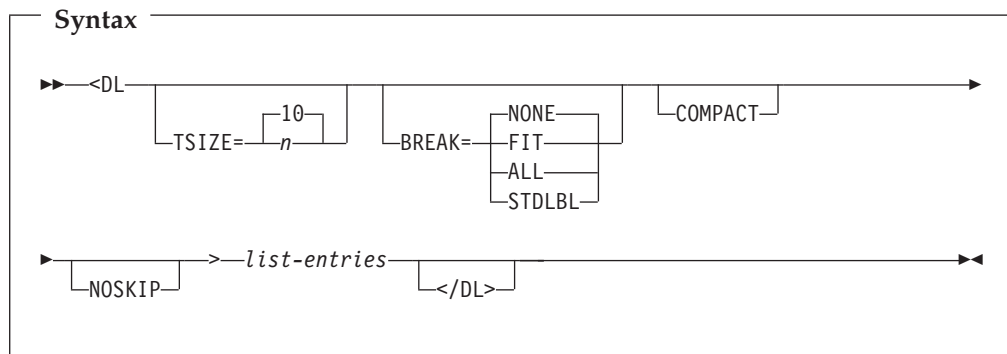
Text that might contain valid tags can be protected from causing formatting problems by using the DATA tag to specify the number of characters that should be treated as text only.

**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 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

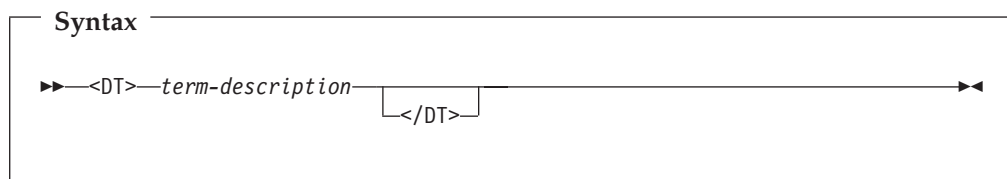
These are the DT and DD tags that make up the list entries.

## Description

In the interactive reanalysis report, the list entries will be presented in white. However, if the STDLBL attribute is used, then the definition terms will be in green.

## DT (definition term)

The DT tag defines a term in a definition list.



**term-description**

This is specified via the DD tag.

**Description**

See “DL (definition list)” on page 437.

**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 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.

A 64-bit address may be specified using an underscore between bits 31 and 32. For example, 1\_00100000. Note that leading zeroes are implied on either side of the underscore to make each address component 32 bits.

**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 will be 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 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.

A 64-bit address may be specified using an underscore between bits 31 and 32. For example, 1\_00100000. Note that leading zeroes are implied on either side of the underscore to make each address component 32 bits.

**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.

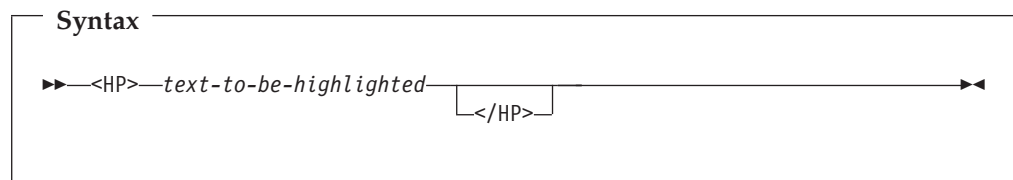
The character-interpreted section on the right-hand side of the display is based on ASCII-encoded hexadecimal values. If the data is known to contain EBCDIC-encoded values, then use the DUMP tag instead (see “DUMP (EBCDIC dump)” on page 439).

In the interactive reanalysis report, the heading text will be 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.



### text-to-be-highlighted

This text displays with highlighted emphasis.

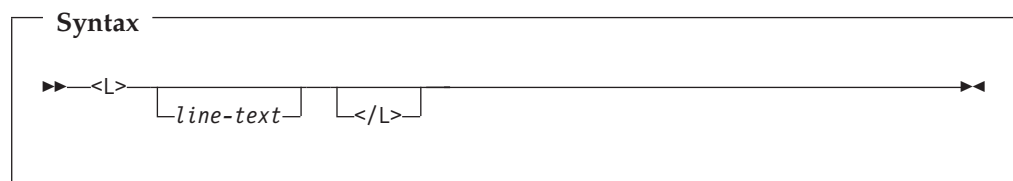
## Description

This tag does not cause a line break.

In the interactive reanalysis report, the text will be presented in turquoise.

## L (line)

The L tag defines a line of text that is not subject to the user preferred display width.



### line-text

This is 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 will then flow onto the following line.

In the interactive reanalysis report, the line text will be presented in white.

**LI (list item)**

The LI tag defines a list item within an note list or an unordered list.

**Syntax**

```

>><LI>—item-text—</LI>

```

**item-text**

This is the item text.

**Description**

In the interactive reanalysis report, the line text will be 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.

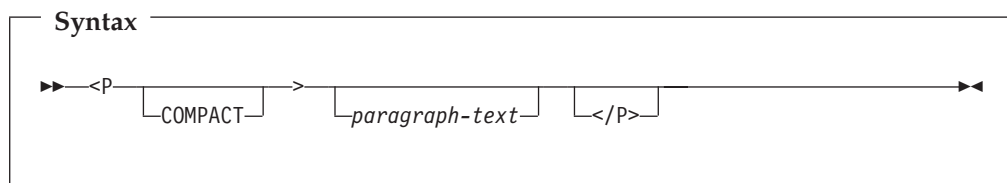
**Description**

This tag causes a line break.

**P (paragraph)**

The P tag defines a paragraph of text.

## P (paragraph)



### COMPACT

This attribute causes the paragraph to format without a blank line before the paragraph.

### paragraph-text

This is 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.

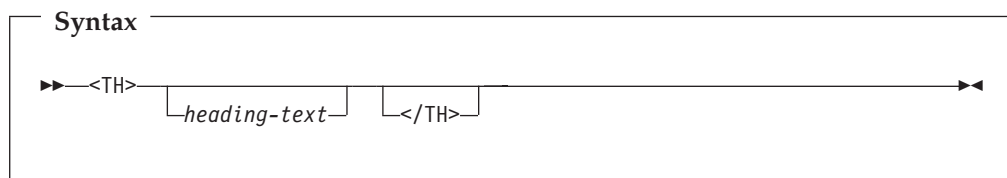
The width of the displayed paragraph is determined by the preferred formatting width specified by the user via the View pull-down menu.

Paragraphs within a list align with the text of the list item.

In the interactive reanalysis report, the paragraph text will be presented in white.

## TH (table heading)

The TH tag defines a table heading line.



### heading-text

This is 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 will then flow onto the following line.

In the interactive reanalysis report, the table heading line text will be 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 will be 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.



---

## 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 is an optional feature.

Refer to “Using the Fault Analyzer plug-in for Eclipse” on page 193 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.

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 can not 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 IDL.SIDISAM1:

```
CONFIG=FA
SPAWN_PROGRAM=IDIGMAIN
SPAWN_REGIONSZ=200
```

## installing the Fault Analyzer plug-in for Eclipse

```
SPAWN_JOBNAME=IDISVRF  
SPAWN_PARM_SECTION  
ISPF_PROF_DSN=&USERID..ISPF.ISPPROF 1  
ISPF_APPL=IDI 2
```

The parameters you might need to change are as follows:

- 1** Change &USERID..ISPF.ISPPROF if a different data set is used for ISPF profile members. &USERID will be replaced by the user ID of the connected user. For example, if the user ID is FRED, and the ISPF application ID in **2** is IDI, then the ISPF profile is retrieved from FRED.ISPF.ISPPROF(IDIPROF).
- 2** 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 may be downloaded to your PC from either

- MVS data set IDL.SIDIDOC2(IDIGUIP).

The downloaded file should be called IDIGUIP.zip and should be transferred in BINARY format.

- From the IBM Problem Determination Tools Plug-ins web site:  
<http://www-01.ibm.com/software/awdtools/deployment/pdtplugins/>

## Install Fault Analyzer plug-in into CICS Explorer

Having downloaded IDIGUIP.zip, 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 called readme.html, which provides installation instructions for the plug-in.

---

## Installing the Fault Analyzer client for IBM Rational Developer for System z

This is an optional feature.

Refer to “Using the Fault Analyzer client for IBM Rational Developer for System z” on page 193 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 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.

## installing the Fault Analyzer client for IBM Rational Developer for System z

Refer to <http://publib.boulder.ibm.com/infocenter/ratdevz/v7r6/index.jsp?topic=/com.ibm.etools.fa.FAClientHelpPlugin/features.html> for additional information about installing this interface.

### Notes:

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 can not 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 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 194 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 will be 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 will vary 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.

## Enabling interactive reanalysis under CICS

1. Add data set IDI.SIDIAUTH to the DFHRPL concatenation of the CICS JCL.
2. Allocate a new profile data set to be assigned to the IPVPROF and IPVTABL 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.
3. Add the following additional DD names to the CICS JCL:

```
//IPVPLIB DD DISP=SHR,DSN=IPV.SIPVPENU25
//          DD DISP=SHR,DSN=IPV.SIPVMENU25
//          DD DISP=SHR,DSN=IDI.SIDIPLIB
//          DD DISP=SHR,DSN=IDI.SIDIMLIB
//          DD DISP=SHR,DSN=IDI.SIDISLIB
//IPVTLIB DD DISP=SHR,DSN=IPV.SIPVTENU25
//          DD DISP=SHR,DSN=IDI.SIDITLIB
//IPVPROF DD DISP=SHR,DSN=IDI.IDIPPROF
```

---

## Installing the Fault Analyzer web browser interface

This is an optional feature.

Refer to “Using the Fault Analyzer web browser interface” on page 194 for information about using this interface.

The Fault Analyzer web browser requires two installations steps:

1. Run the sample job to create a program object in a z/OS Unix System Services file
2. Make the required updates to the HTTP server configuration files.

These steps are described more detailed in the following.

### Running the sample job to create a program object in a z/OS Unix System Services file

The sample job, IDISBRWS, which is located in the IDI.SIDISAM1 data set, uses the binder to create a program object in a z/OS Unix System Services file. Before running this job, you need to make any necessary changes, as described in the comment section of the job itself.

Having made the necessary alterations the job should complete with a zero return code.

### Making the required updates to the HTTP server configuration files

The following Service statements need to be added to the configuration file of your HTTP server:

```
Service /FAULTANAL/*    /#USSPATH/idipbrws.so:idiShowDirectory/*
Service /FAULTHIST/*    /#USSPATH/idipbrws.so:idiShowReport/*
Service /FAULTHDSEL/*    /#USSPATH/idipbrws.so:idiSelectHeader/*
```

The /#USSPATH/ needs to be changed to the same value used in job IDISBRWS described above. This is only necessary if the path used is not in the LIBPATH statement of the HTTP server. If the path is in the LIBPATH, then the /#USSPATH/ can be removed.

---

25. Data set created as part of the IBM Problem Determination Tools Common Component (PDTCC) installation.

---

## Part 3. Fault Analyzer reference information

<b>Chapter 33. Options</b> . . . . .	451	StorageRange . . . . .	498
Where to specify options . . . . .	452	SystemWidePreferred. . . . .	499
Parmlib member IDICNF00. . . . .	453	UseIDISTime . . . . .	501
User-options module IDICNFUM. . . . .	453	<b>Chapter 34. Data areas</b> . . . . .	503
Configuration-options module IDIOPTLM. . . . .	454	Non-REXX user exit buffered data format . . . . .	503
The _IDI_OPTSFILE environment variable. . . . .	454	Data area descriptions . . . . .	505
User options file IDIOPTS . . . . .	454	CTL - Analysis Control user exit parameter list	505
The JCL EXEC statement PARM field . . . . .	455	ENV - Common exit environment information	512
The _IDI_OPTS environment variable . . . . .	455	EPC - End Processing user exit parameter list	519
Option descriptions . . . . .	455	LST - Compiler Listing Read user exit	
AdditionalIDIOffDD . . . . .	455	parameter list . . . . .	520
CICSDumpTableExclude. . . . .	456	NFY - Notification user exit parameter list. . . . .	522
CICSTraceMax . . . . .	456	UFM - Formatting user exit parameter list. . . . .	523
DataSets . . . . .	458	UTL - IDIUTIL Batch Utility user exit parameter	
Data set logical replacement or concatenation		list . . . . .	529
order . . . . .	460	XPL - Message and Abend Code Explanation	
Dropping IDICNF00 parmli b member data		user exit parameter list . . . . .	530
set specifications . . . . .	460	<b>Chapter 35. Return codes</b> . . . . .	533
Data set name substitution symbols . . . . .	461	Batch reanalysis (IDIDA) . . . . .	533
DeferredReport. . . . .	463	IDIUTIL batch utility . . . . .	533
Detail . . . . .	465	IDILANGX . . . . .	533
DumpDSN . . . . .	466	<b>Chapter 36. Messages</b> . . . . .	535
DumpRegistrationExits . . . . .	466	Fault Analyzer messages . . . . .	535
ErrorHandler . . . . .	467	IDILANGX messages. . . . .	556
Exclude/Include . . . . .	468		
Using Exclude/Include options with dump			
registration processing . . . . .	471		
Exclude/Include trace information . . . . .	471		
Exclude/Include wildcard examples. . . . .	472		
Exits . . . . .	473		
Dropping IDICNF00 parmli b member user			
exit specifications . . . . .	475		
FaultID . . . . .	475		
GenerateSavedReport. . . . .	476		
HistCols . . . . .	477		
Include . . . . .	478		
InteractiveExitPromptSeconds . . . . .	478		
Language. . . . .	479		
Locale . . . . .	480		
LoopProtection . . . . .	480		
MaxMinidumpPages . . . . .	481		
NoDup . . . . .	482		
NoDup(ImageFast(...)) . . . . .	483		
NoDup(CICSFAST(...)) . . . . .	486		
NoDup(NORMAL(...)) . . . . .	489		
PermitLangx. . . . .	490		
PreferredFormattingWidth . . . . .	491		
PrintInactiveCOBOL . . . . .	492		
Quiet . . . . .	492		
RDZClient . . . . .	493		
RefreshExits . . . . .	493		
RetainCICSDump . . . . .	495		
RetainDump. . . . .	495		
Source. . . . .	496		
SpinIDIREPRT . . . . .	497		
StoragePrintLimit . . . . .	497		





---

## 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 may 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 243.
3. Configuration-options module IDIOPTLM.  
For more information, see “Configuration-options module IDIOPTLM” on page 454.
4. Options located via an IDICNFUM user-options module.  
This 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 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 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 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 377.

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 402.

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 through 71 are processed.
- Options may 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 may 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 may 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 89 and “Interactive reanalysis options” on page 97 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 377.

## 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 267.

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 125 on page 268.

## 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 will try to open the data set members in the order of their specification. The first data set and member found to be available will be 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 allocated to the JOBLIB DDname, this effectively provides job-level control of default options.

The load module may 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 a 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 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 wish 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 324 for an alternative method of activating this trace under CICS.)

The name of the selected user-options module data set and member will be saved by Fault Analyzer in the history file and automatically reused as the default

## Where to specify options

options file during reanalysis of the fault. For batch reanalysis, the data set and member will be included in the generated JCL with the IDIBOPT DDname.

## Configuration-options module IDIOPTLM

As an alternative to several of the SMP/E USERMODs described in Chapter 16, “Customize Fault Analyzer by using USERMODs,” on page 243, you can create an configuration-options load module. This can reduce the effort required to maintain Fault Analyzer, since SMP/E installed USERMODs occasionally must be restored and re-applied in order to install SMP/E maintenance.

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 LNKST so that the IDIOPTLM load module can be found by all jobs. IBM recommends the use of IDI.SIDIAUTH for this authorized library.

A sample job to create an IDIOPTLM configuration-options load module is provided as member IDIOPTLM in data set IDI.SIDISAM1.

## 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 a 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 through 71.

- **HFS path and file name specification**

If specifying a 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 will be 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 328).

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
//SYSMDUMP DD DISP=SHR,DSN=MY.DUMP.DATA.SET
//IDIOPTS DD *
    RetainDump(ALL) /* do not suppress the dump if MYAPPL
                      abends */
/*
```

Figure 165. 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 through 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 will be 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:

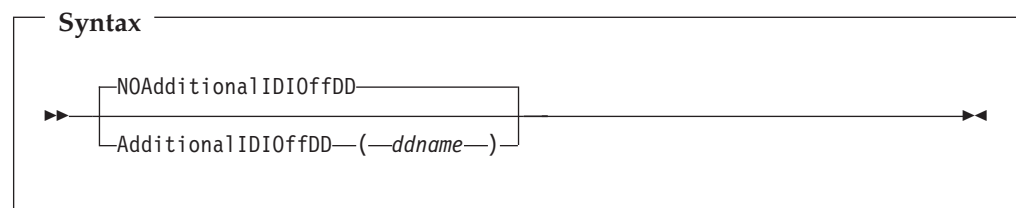
```
/* Sample BPXBATCH job
//RUN EXEC PGM=BPXBATCH, ...
/*
//STDENV DD *
    _IDI_OPTS=DataSets(IDIHIST(MY.HIST)),Detail(L)
/*
```

---

## Option descriptions

The following explains each option in detail.

### AdditionalIDIOffDD



## AdditionalIDIOffDD

The AdditionalIDIOffDD option can be used to provide an additional 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 368). If specified, any job with *ddname* allocated will 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 will 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 233).

This option is not included in the section of the fault analysis report that shows options in effect.

## CICSDumpTableExclude

### Syntax



The CICSDumpTableExclude option will exclude 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.

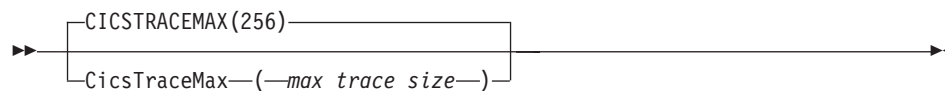
Exclusion of Fault Analyzer processing based on this option precedes any other Fault Analyzer methods of preventing analysis, such as the Exclude or NoDup options (see “Real-time exclusion processing” on page 25).

When NOCICSDumpTableExclude is in effect (default), then Fault Analyzer will 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

### Syntax



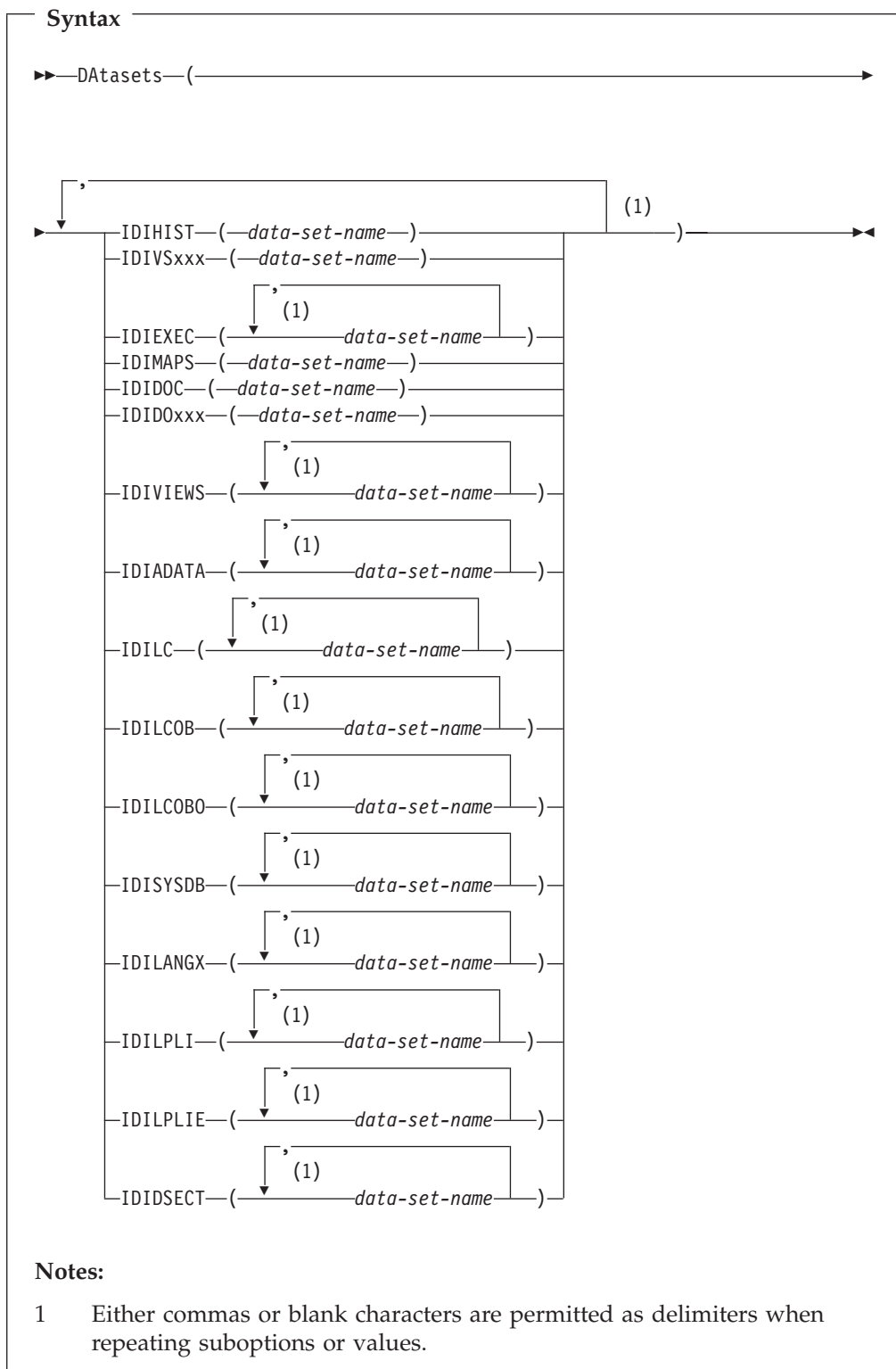
The CICSTraceMax option is applicable to CICS system dump analysis only, and is used to specify the maximum CICS trace table size in kilobytes which may be included in a fault entry minidump.

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 will be read from the associated CICS system dump during fault entry reanalysis.

## DataSets



The DataSets option specifies as suboptions the DDnames and their associated data set names that are to be dynamically allocated by Fault Analyzer.



<b>IDIHIST</b>	The name of the PDS(E) history file where the fault entry is to be or was written. Default value is IDI.HIST.
<b>IDIEEXEC</b>	The name of one or more PDS(E) data sets containing REXX user exits.
<b>IDIMAPS</b>	The name of the PDS(E) data set containing data area mappings provided with Fault Analyzer. Default value is IDI.SIDIMAPS.
<b>IDIDOC</b>	The name of the distributed book index and override PDS(E) data set. Default value is IDI.SIDIDOC1.
<b>IDIDOxxx</b>	The name of an additional 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.
<b>IDIVSxxx</b>	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.
<b>IDIVIEWS</b>	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.
<b>IDIADATA</b>	The name of one or more sequential or PDS(E) data sets holding Assembler SYSADATA files.
<b>IDILC</b>	The name of one or more sequential or PDS(E) data sets holding C compiler listings.
<b>IDILCOB</b>	The name of one or more sequential or PDS(E) data sets holding COBOL compiler listings (other than OS/VS COBOL).
<b>IDILCOBO</b>	The name of one or more sequential or PDS(E) data sets holding OS/VS COBOL compiler listings.
<b>IDISYSDB</b>	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 CDADBGLD utility.)
<b>IDILANGX</b>	The name of one or more sequential or PDS(E) data sets holding side files.
<b>IDILPLI</b>	The name of one or more sequential or PDS(E) data sets holding PL/I compiler listings (other than Enterprise PL/I).
<b>IDILPLIE</b>	The name of one or more sequential or PDS(E) data sets holding Enterprise PL/I compiler listings.
<b>IDIDSECT</b>	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 145.

IDIHIST, IDIEEXEC, 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 377 for details).
2. Any explicitly coded JCL statements for the DDname.
3. Data sets from all DataSets options specified in the JCL EXEC statement PARM field (reanalysis only).
4. Data sets from all DataSets options specified in the user options file.
5. Data sets from all DataSets options specified in the parmlib configuration member.

This includes the use of a system-wide alternate parmlib data set (for details, refer to “Parmlib member IDICNFxx” on page 267.) 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 specified with the DataSets option may contain standard MVS symbols<sup>26</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 may be used as part of the data set name(s) for each DDname:

Table 26. Permitted data set name substitution symbols by DDname

DDname	&PGM.	&SYSUID.	&TSOPFX.	&USERID.	&SYSNAME.
IDIHIST					X
IDIVSxxx					X
IDIEEXEC				X (Note 1)	X
IDIMAPS					X
IDIDOC					X
IDIDOxxx					X
IDIVIEWS (Note 2)		X	X		X
IDIADATA	X			X	X
IDILC	X			X	X
IDILCOB	X			X	X
IDILCOBO	X			X	X
IDISYSDB	X			X	X
IDILANGX	X			X	X
IDILPLI	X			X	X
IDILPLIE	X			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 59).
- 2 These symbols may be used in the data set names specified in the IDIVIEWS suboption of the DataSets option, as well as in the data set names specified in the individual view members contained within the IDIVIEW data sets—for details, see "Setting up views" on page 250.

If, at the time of performing symbol value substitution, a value for a symbol is unavailable, then any data set names which include that symbol will be ignored. An example of this is the &USERID. variable mentioned in the preceding note.

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:

26. 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

**&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 utilize 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 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, if the option

```
DataSets(IDIVIEWS(&SYSUID..VIEWS,PROD.VIEWS))
```

is specified 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 to 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 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, if the option

```
DataSets(IDIVIEWS(&TSOPFX..VIEWS,PROD.VIEWS))
```

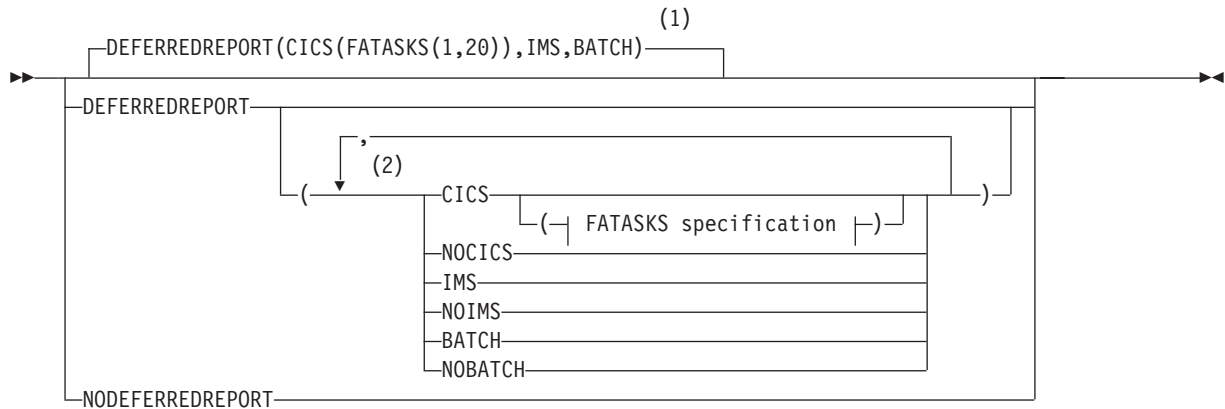
is specified in the IDICNF00 parmlib member, then all users of the Fault Analyzer ISPF interface for whom a data set named *tso-prefix.VIEWS* exist, where *tso-prefix* is the user's TSO profile prefix, can place private Fault Analyzer View definitions in this data set to either complement any installation-wide 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.

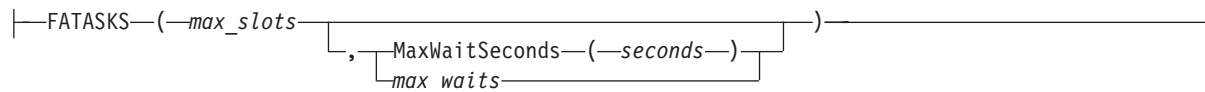
**&SYSNAME.** All instances of this symbol are substituted by the system name from CVTSNAME, as at the time of substitution.

## DeferredReport

### Syntax



### FATASKS specification:



### Notes:

- 1 The product default, when no DeferredReport option has been specified, is `DEFERREREDREPORT (CICS (FATASKS (1,20)), NOIMS, NOBATCH)`.
- 2 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

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 (CICS FATasks (max_slots<,max_waits>))`

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 will be 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)`

## DeferredReport

will disable the DeferredReport option for such an application, whereas  
DeferredReport(CICS,NoIMS)

will enable it.

When used with CICS, the optional FATASKS suboption can be specified:

### *max\_slots*

Specifies the maximum number of execution slots that will be 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 from 1 to 6.

The default is 1.

If NODeferredReport is specified, then the maximum number of execution slots will be 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 from 0 to 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 an additional 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 from 1 to 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 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.

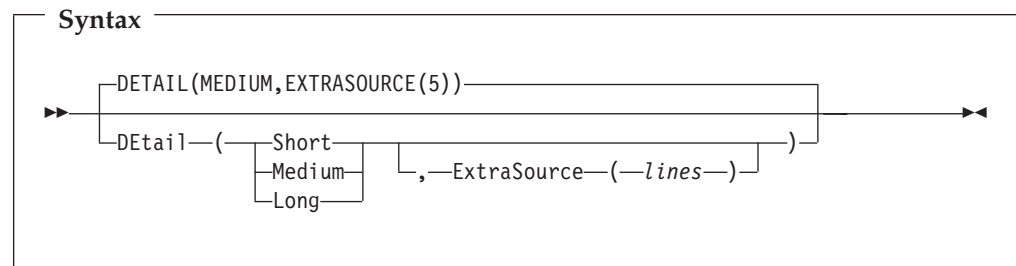
### **Notes:**

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 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 will be 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 377.
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 234.

## 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

Depending on the specification of the Detail option, all or parts of the complete report are produced:

- If Detail(SHORT) is specified, all sections of the report are included, except for the system-wide information. Detailed information is only included for the point of failure event.
- If Detail(MEDIUM) is specified, all sections of the report are included. However, detailed information is only included for events up until (and including) the firstabend. This is the default.
- If Detail(LONG) is specified, all sections of the report are included and detailed information is provided for all events.

## DumpDSN

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 wish 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 will be 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 will contain ASA printer control characters.

## DumpRegistrationExits

### Syntax

```
►►—DumpRegistrationExits—(—(1)—CONTROL—(—Exit name specification—)—)————►►  
                        |  
                        NOTIFY
```

### Exit name specification:

```
      |  
      (1)  
      |  
      | exit_name  
      |  
      |  
      | REXX—(—(1)—exit_name—)—  
      |
```

### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

The DumpRegistrationExits option specifies the types and names of user exits to be invoked during dump registration processing (IDIXTSEL). Any number of exit names can be specified for a given exit type, and all exits will be attempted invoked.



Multiple specifications of the DumpRegistrationExits option are cumulative.

Exits may 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:

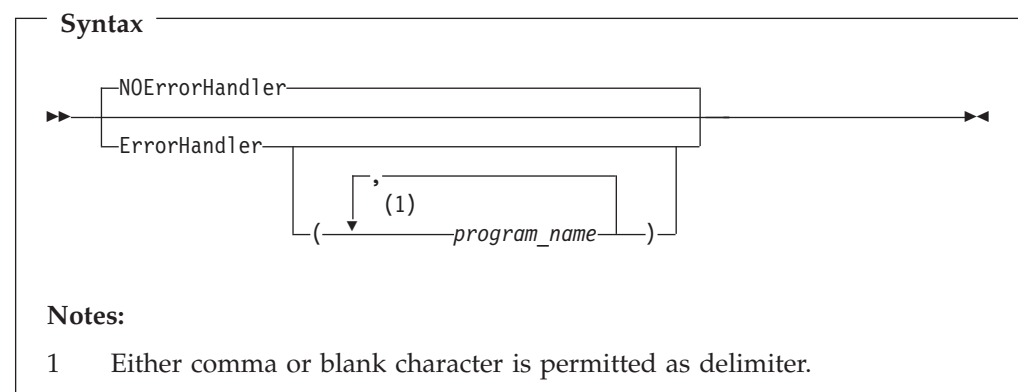
- CONTROL** Analysis Control user exit. This exit can be used to modify options in effect. For details, see “Analysis Control user exit (Dump registration)” on page 380.
- NOTIFY** Notification user exit. This exit can, for example, be used to provide installation-specific notification of recorded faults. For details, see “Notification user exit (Dump registration)” on page 411.

The exit name specified as *exit\_name* may 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 will not be invoked and will cause 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.

The 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 will be ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS region or batch job.

## ErrorHandler



## ErrorHandler

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 will become the point-of-failure event, and thus be used in duplicate fault determination, etc.

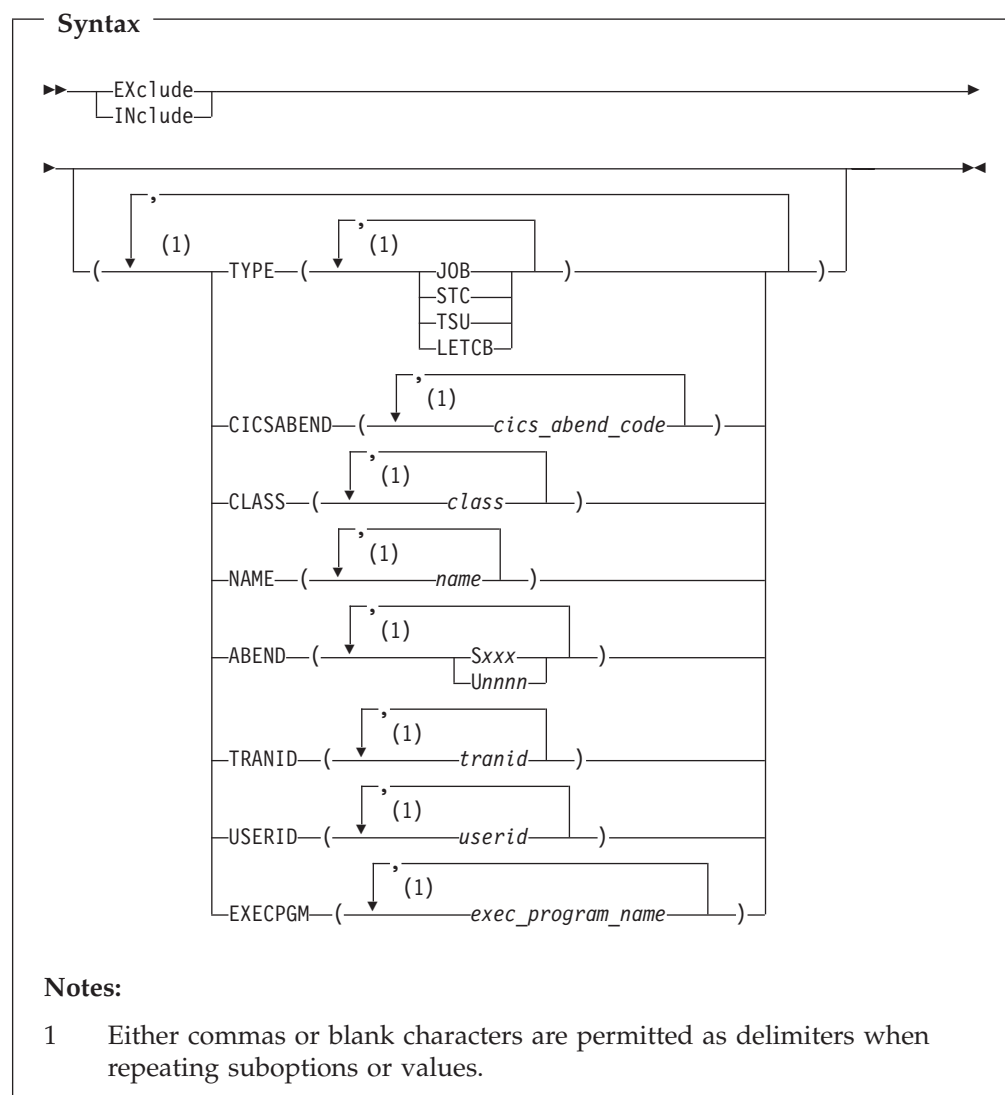
**Note:** If no earlier user-code event can be determined, then it is possible that the error handler program event is still designated as the point of failure.

The last specification of the ErrorHandler or NOErrorHandler option will be in effect. Subsequent specifications will 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 269, 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.



where:

- TYPE** Specifies the type of work unit as either JOB (batch jobs), STC (started tasks), or TSU (TSO users).
- An additional type, LETCB, specifies that LE must be active for the abend TCB. This 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 alphanumeric characters.
- The abend code tested is the final abend code for the transaction.
- 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*.

## Exclude/Include

<b>ABEND</b>	<p>Specifies one or more system or user abend codes as one of the following:</p> <p><b>Sxxx</b> where <i>xxx</i> is a three-character hexadecimal system abend code (for example, S0C4)</p> <p><b>Unnnn</b> where <i>nnnn</i> is a four-character decimal digit user abend code (for example, U4039)</p> <p>The abend code tested is the final abend code for the abending job step.</p>
<b>TRANID</b>	Specifies the names of one or more CICS transactions as <i>tranid</i> .
<b>USERID</b>	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 <i>userid</i> .
<b>EXECPGM</b>	Specifies the program name from the JCL EXEC statement PGM keyword, as <i>exec_program_name</i> .

**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, may 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 will never be met, for example  
`Exclude(TYPE(STC) CLASS(A))`

The reason why this will never be 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 will analyze any abends that occur.

- If more than one *type* (JOB, STC, TSU, or LETCB), *cics-abend-code*, *class*, *name*, *abend-code* (Sxxx or Unnnn), *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 additional information, see “Controlling which jobs are analyzed with Exclude processing” on page 269.

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 368.

### 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 is due to the dump being written by the dump services address space (for details, see “Dump registration processing” on page 24), 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 (Dump registration)” on page 380) is 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 512).

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 324 for an alternative method of activating this trace under CICS.)

The trace information will be written to the IDITRACE DDname destination. An example of an Exclude/Include option-processing trace follows:

## Exclude/Include

---

EXCLUDE/INCLUDE option criterion match values from abending job:

ABEND. . . . . : S0CB  
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 166. Sample Exclude/Include option-processing trace*

Trace information is not available for CICS transactions.

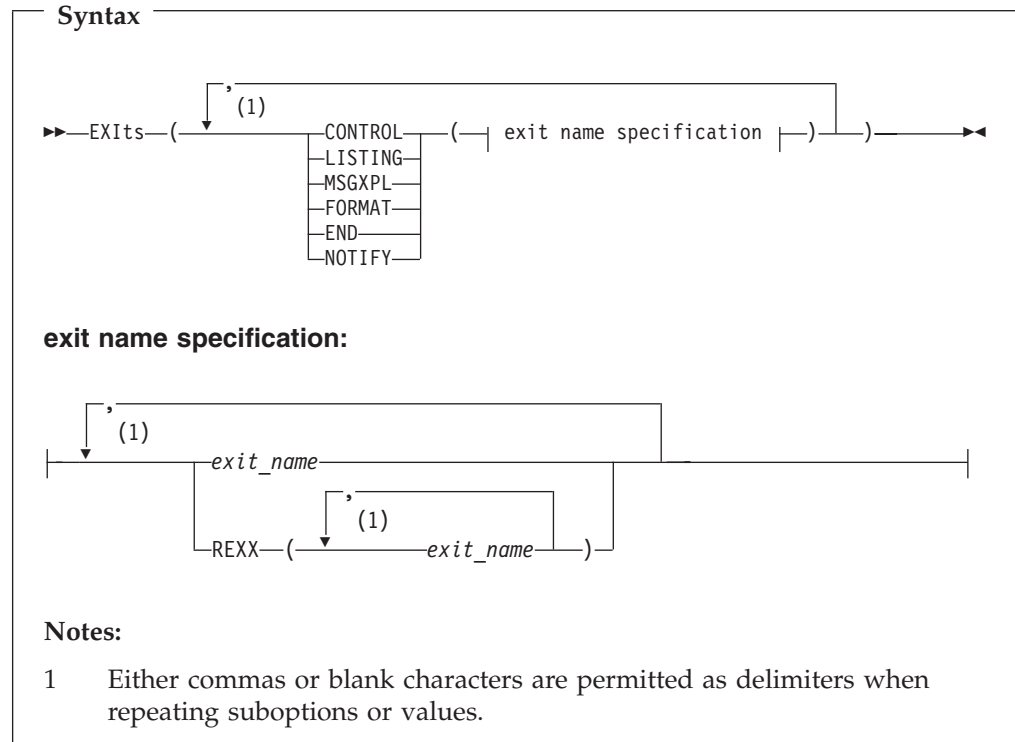
### Exclude/Include wildcard examples

The following are examples of Exclude/Include criterion values with wildcards:

*Table 27. 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 will be attempted invoked.

**Note:** For information about specifying user exits for the IDIUTIL batch utility, see “EXITS control statement” on page 359 instead.

Multiple specifications of the Exits option are cumulative.

Exits may 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 or exclude a fault from analysis. For details, see “Analysis Control user exit” on page 377.

**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 contained in available MVS PDS (or PDSE) data sets. For details, see “Compiler Listing Read user exit” on page 382.

## Exits

<b>MSGXPL</b>	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 386.
<b>FORMAT</b>	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 390.
<b>END</b>	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 402.
<b>NOTIFY</b>	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 405.

The exit name specified as *exit\_name* may be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

<b>NONE</b>	The special name 'NONE' represents a 'null' exit that will not be invoked and will cause further attempts to invoke exits of the specified type to be terminated.
<b>-DROPCNF-</b>	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 will be written to the “Options in Effect” section of the analysis report. In this section, all specified exits will be listed, and those of each type that were invoked (if any) will be 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:

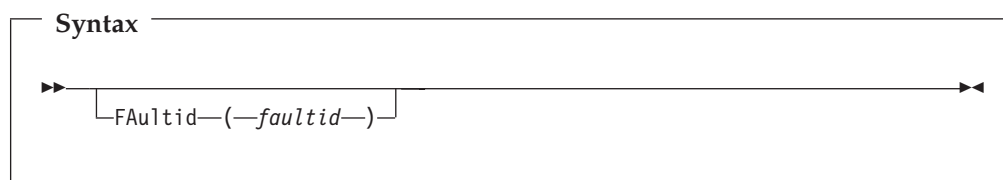
```
Exits(Control(FRED3,-DROPCNF-,FRED4))
```

then the final logical concatenation of Analysis Control user exits would be:

Type	Name
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.

## 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:

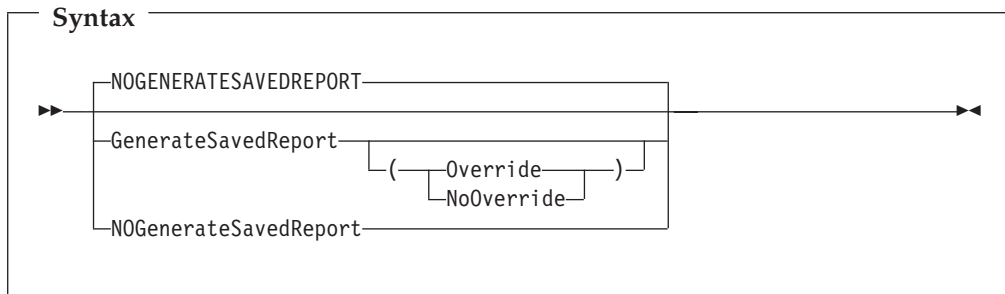
*cccnnnnn*

where *ccc* is the assigned history file prefix (1 to 3 characters) and *nnnnn* is a 5-digit decimal fault number.

#### Notes:

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 will be automatic.

## GenerateSavedReport



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 will be generated, regardless of whether one already exists or not.

If the NoOverride suboption is used, then a saved report will be generated only if one does not already exist. This is the default if the GenerateSavedReport option is specified without a suboption.

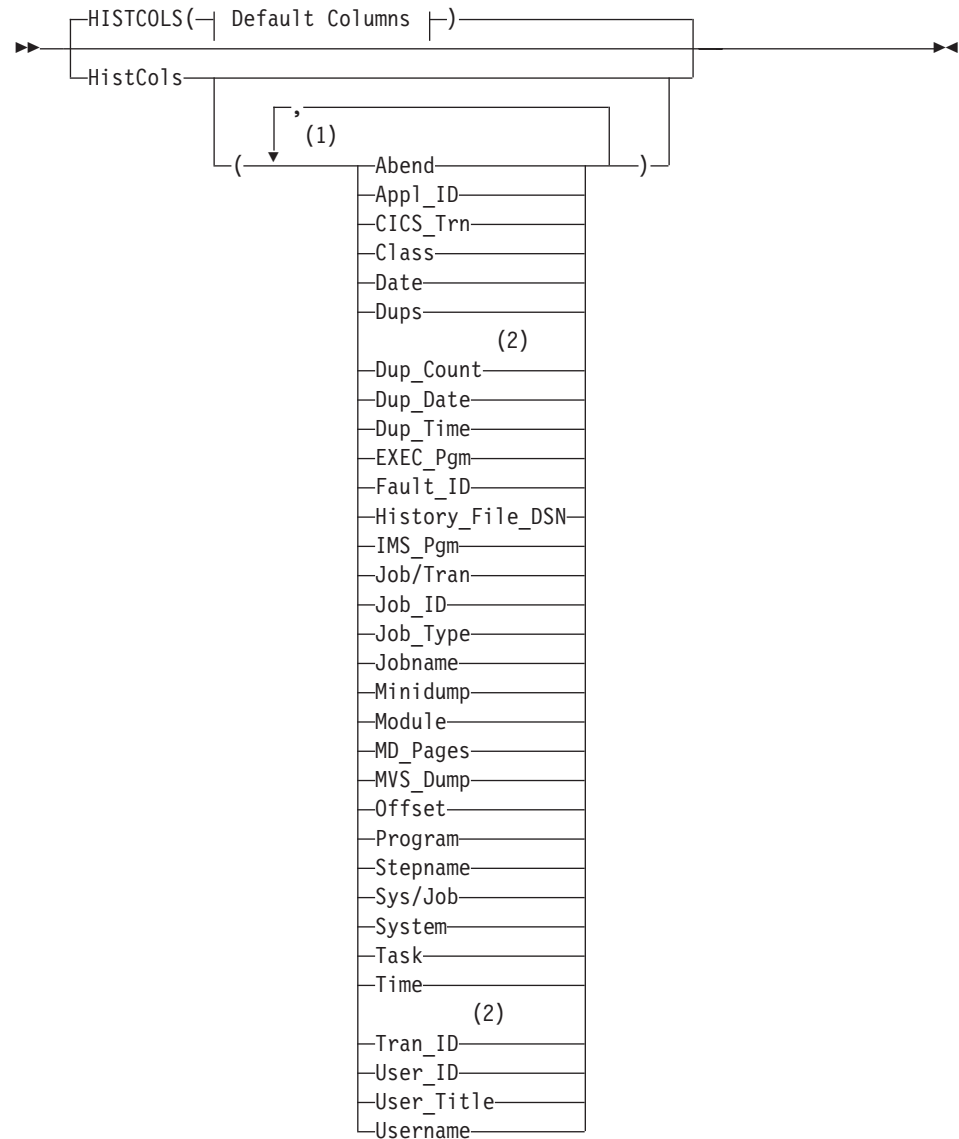
If the GenerateSavedReport option is in effect, then source information data sets (compiler listings or side files) used during real-time processing are automatically included during the reanalysis.

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 410.

If the GenerateSavedReport option is used with batch analysis of an MVS dump data set specified using the DumpDSN option, then this option will cause 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 will be issued.

## HistCols

### Syntax



### Default Columns:

| Fault\_ID | Job/Tran | User\_ID | Sys/Job | Abend | Date | Time |

### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.
- 2 Deprecated column name—refer to “Fault entry list column configuration” on page 41 for details.

The HistCols option can be used to customize the columns of information 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 which 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.

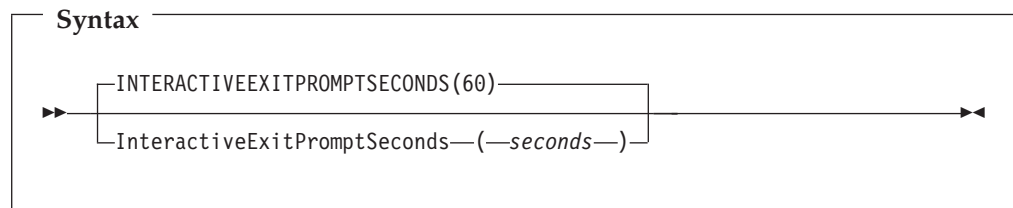
For additional information about the customization of the Fault Entry List display, and for an explanation of the individual columns that can be specified using the HistCols option, see “Fault entry list column configuration” on page 41.

This option is not included in the section of the fault analysis report that shows options in effect.

## Include

See “Exclude/Include” on page 468.

## InteractiveExitPromptSeconds



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 will be shown. Otherwise, the user will be requested to press the Enter key to confirm exit from the interactive report.

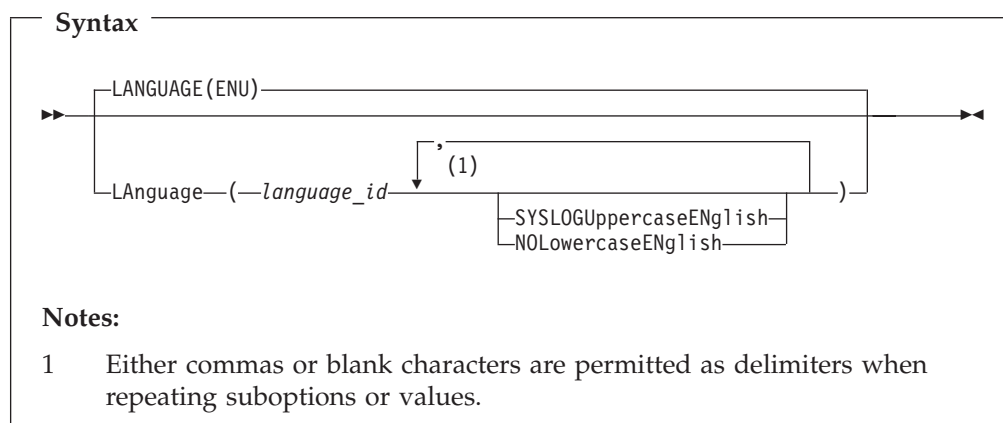
This option does not apply to real-time analysis or batch reanalysis.

This option is not included in the section of the fault analysis report that shows options in effect.

The valid *seconds* range is from 0 to 99999, both inclusive:

- If 0 is specified, then the prompt will be displayed unconditionally.
- If 99999 is specified, then the prompt will never be displayed.

## 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 has been installed)
KOR	Korean (available only if the Korean feature of Fault Analyzer has been installed)

The following optional suboptions may be specified:

### SyslogUppercaseEnglish

Where possible, causes all WTO messages to be in uppercase English only.

This suboption may be abbreviated to SYSLOGUEN.

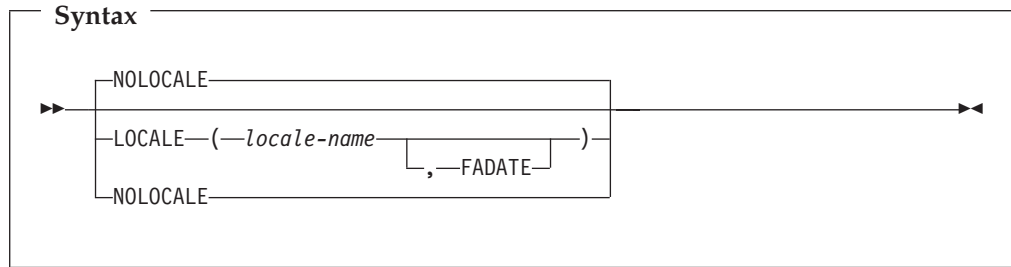
### NoLowercaseEnglish

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 'KANJI' mode to prevent lowercase English characters from being unreadable.

This suboption may be abbreviated to NOLEN.

If the Language option is specified in the IDICNF00 parmlib configuration member, it should be the first option specified. This will permit the maximum number of language-dependant messages to be issued by Fault Analyzer.

## Locale

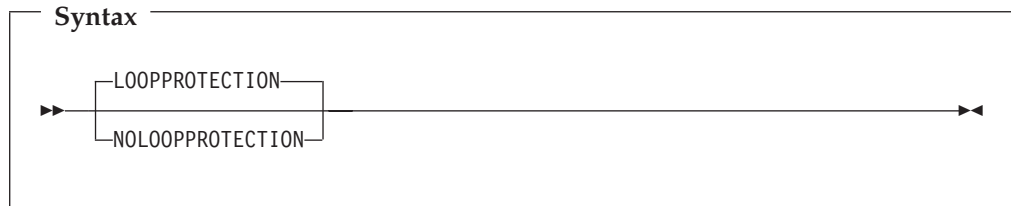


The Locale option specifies the locale to be used for cultural environment-dependent presentation. Affected are things like date and time formatting, collating sequences of sorted information, and determination of non-printable characters.

If the optional FADATE suboption is specified, then all dates presented will be 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).

## LoopProtection



This option is used to activate or deactivate the Fault Analyzer loop/wait protection feature. With this feature active (this is the default), Fault Analyzer will terminate processing if the maximum expected elapsed time for any one of several internal checkpoints has been exceeded. If this happens, then message IDI0092S will be issued.

A user exit can deactivate the loop/wait protection feature by setting the ENV.LOOPPROTECTION\_OPT field to N. For additional information about this field, see “ENV - Common exit environment information” on page 512.

**Note:** Even with the NoLoopProtection option in effect, there is no guarantee that an analysis, that might otherwise be terminated with the IDI0092S message, will successfully complete. This is because the job might still be subject to normal installation-imposed MVS execution time limits, or actually be in a never-ending loop or wait condition, that will eventually require the job to be canceled.

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

### Syntax

```

  MAXMINIDUMPPAGES(4096)
  MaxMinidumpPages(—max_pages—)

```

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 will be saved as a 'minidump' against which reanalysis can be performed. If the number of referenced pages exceed *max\_pages*, then no minidump will be saved.

**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.

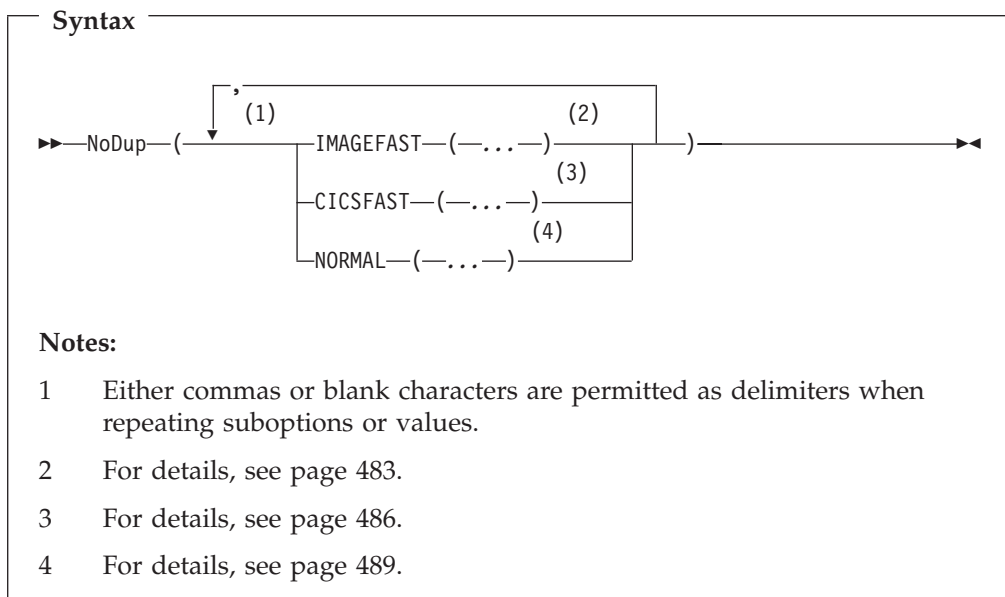
This option does not apply to batch or interactive fault reanalysis.

The section of the fault analysis report that shows options in effect will include this option and further indicate if the limit was exceeded, and if so, by how many pages. If it was exceeded, then message IDI0052I will also be issued.

The valid *max\_pages* range is from 0 to 524288, both inclusive.

**Note:** The use of an Analysis Control or End Processing user exit may effectively override the MaxMinidumpPages option in effect.

## NoDup



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 483).  
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 will have its duplicate count incremented accordingly.

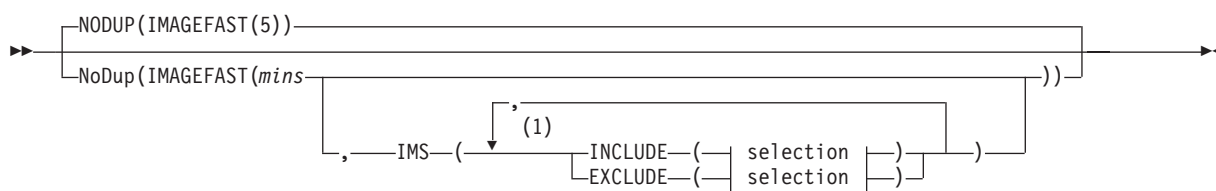
This option applies to the real-time analysis only.

If NoDup(NORMAL(*hours*, ...)) is in effect, with a non-zero value of *hours*, then this option will be 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, will be provided. NoDup(CICSFAST) or NoDup(ImageFast) option information will not be included, since no report is written if the associated duplicate criteria matched.

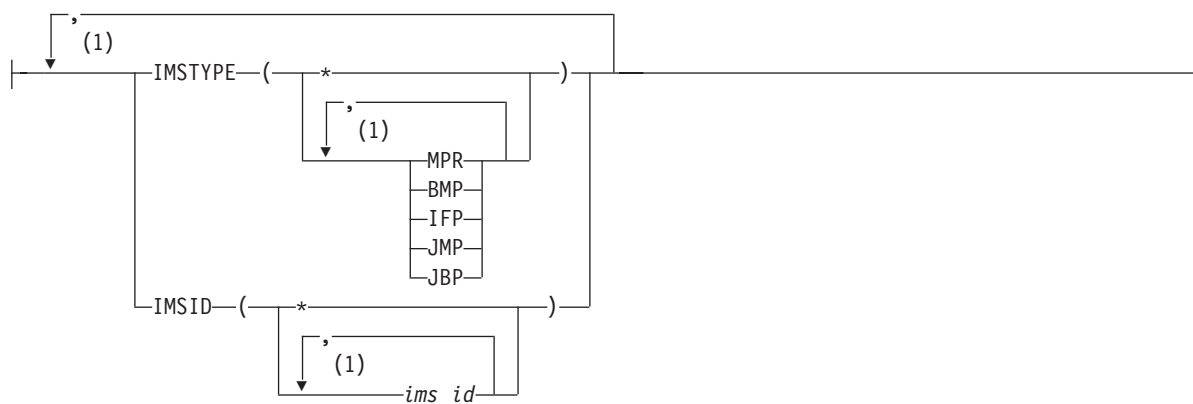
While the syntax for each duplicate sub-type is shown separately in the following, they can be specified together in a single NoDup option if desired.

### NoDup(ImageFast(...))

#### Syntax



#### selection:



#### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

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* will be 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".

## NoDup

**Note:** In order to enable IMS fast duplicate fault suppression, the Fault Analyzer IDIS 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 233.

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 from 0 to 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 is the equivalent of having specified the option

```
NoDup(ImageFast(minutes,IMS(INCLUDE(IMSTYPE(*),IMSID(*)))))
```

However, if only certain IMS jobs are to be eligible for IMS fast duplicate fault suppression, then this can be controlled using the INCLUDE or EXCLUDE suboptions of the NoDup(ImageFast(*minutes*,IMS(...))) option, given the following specification rules:

- An IMSTYPE criterion will match if the specified values include the IMS region type 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 will match, and the current fault deemed ineligible for IMS fast duplicate fault suppression.

- An IMSID criterion will match 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 will match, 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 criterions of the specified suboption match, will change the eligibility state accordingly.

For example, if the following option is specified:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR,BMP),IMSID(ABC1,ABC2,ABC3)))))
```

and the current fault is an IMS BMP region with subsystem ID ABC4, then the IMSTYPE criterion will match, but the IMSID criterion will 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* will take effect.

- A wildcard (\*) may 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 on page 451. For example, the IDICNF00 parmlib member is read prior to any IDIOPTS user options file used.
- The most generic criteria should be specified first, followed by more specific ones.

For example, if an installation wishes to utilize IMS fast duplicate fault suppression for only IMS MPR jobs using subsystem IDs other than ABC1, then specifying the following options would achieve this:

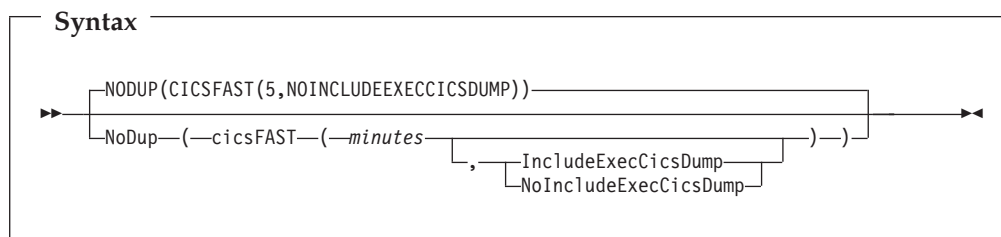
```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(*) /* Exclude everything */
                           INCLUDE(IMSTYPE(MPR)) /* Include only IMS MPR regions */
                           EXCLUDE(IMSID(ABC1)) /* Exclude subsystem ID ABC1 */
                           )))
```

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(ImageFast(...)) option in effect, the fault analysis will be skipped, the duplicate count associated with the original fault is incremented by one, and message IDI0121I is issued.

## NoDup

**Changing the NoDup(ImageFast) option:** The NoDup(ImageFast) option is used by the Fault Analyzer IDIS subsystem only. Changes to the NoDup(ImageFast) option will 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 233).

## NoDup(CICSFAST(...))



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 will be 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 167 on page 487, 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”

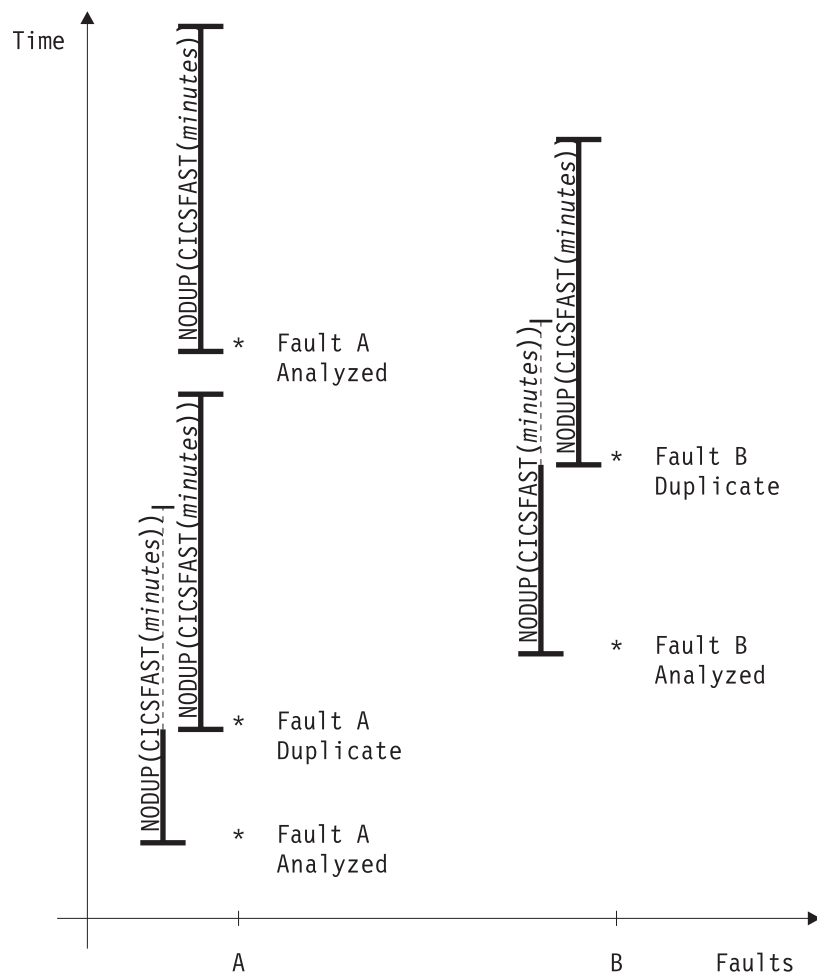


Figure 167. NoDup(CICSFAST) illustration

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 from 0 to 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.

## NoDup

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 will also be performed when using the EXEC CICS DUMP command:

- If a TACB is available, then the duplicate determination will be 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 will be performed on the basis of:
  - Transaction IDs
  - CICS dump codes
  - Failing program names

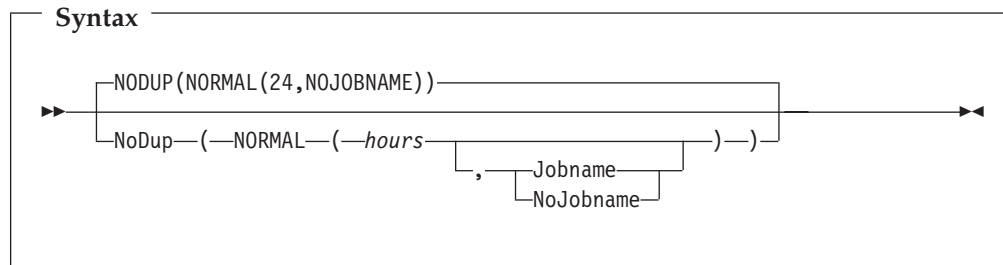
When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(CICSFAST(...)) option in effect, the writing of the history file entry will be suppressed, the duplicate count associated with the last fault is incremented by one, and message IDI0066I is issued. This type 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 326.

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 324.

**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 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 will be reflected down to the exit for NoDup(CICSFAST) action. This means either waiting for another unique fault (this can be forced by creating a new abend with CECI to perform an EXEC CICS ABEND) or the NoDup(CICSFAST) time-out to occur.

## NoDup(NORMAL(...))



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 is the standard duplicate fault detection mechanism used, for example, for abending batch jobs.

The valid range of *hours* is from 0 to 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 will not be 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 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=*
```

## NoDup

(See “IDITRACE under CICS” on page 324 for an alternative method of activating this trace under CICS.)

**Note:** Only NoDup(NORMAL) trace information is available, not NoDup(CICSFAST).

The trace information will be written to the IDITRACE DDname destination. An example of a NoDup option-processing trace follows:

---

```
NoDup(NORMAL(24)) option processing match values:
  Abend Code . . . . . : S0C7
  CICS Transaction ID. . . . . : n/a
  Module Name. . . . . : IDISCB1
  CSECT Name . . . . . : IDISCB1
  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
```

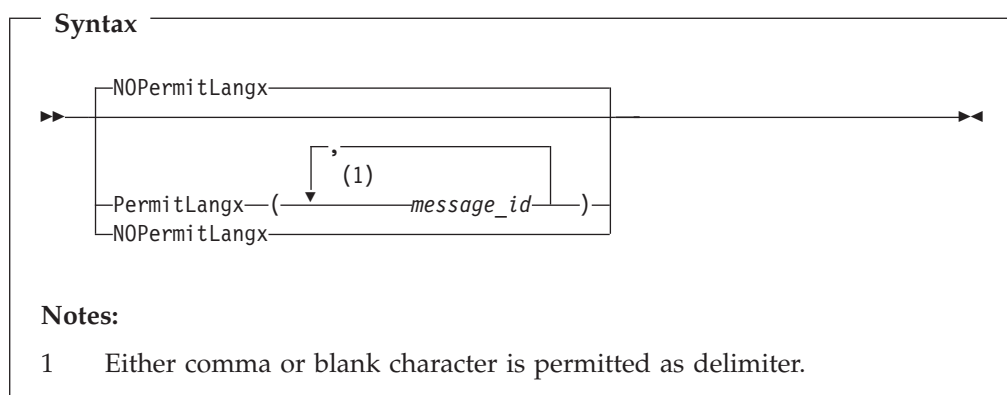
---

*Figure 168. 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

## PermitLangx



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 which 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. For example:

```
PermitLangx(IEL0490I)
```

The PermitLangx option is applicable to COBOL or PL/I compiler listings only.



To specify that all messages of a given severity should be permitted, the following notation can be used:

`XXX-C`

where

`XXX` is the message prefix.

`C` is 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 IEL0490I, the compiler listing might show:

IEL0490I E 233 INVALID USE OF 'DEFINED' IN DECLARATION OF 'WZ12211'.

In this case, the message severity level is E, which generally implies a return code of 8.

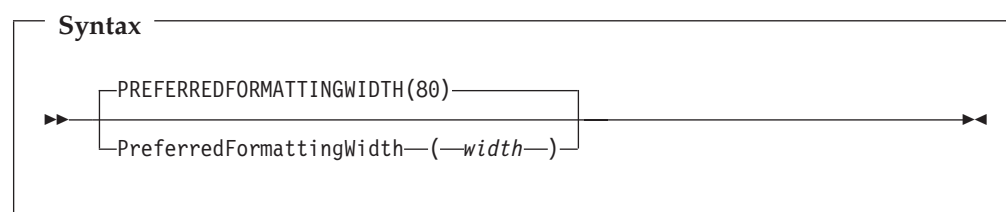
Only the last specification of the PermitLangx option is used. That is, multiple specifications are not cumulative.

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. For example, the normalized length of

`PermitLangx(FRED1 FRED2,FRED3 )`

is 17.

## PreferredFormattingWidth



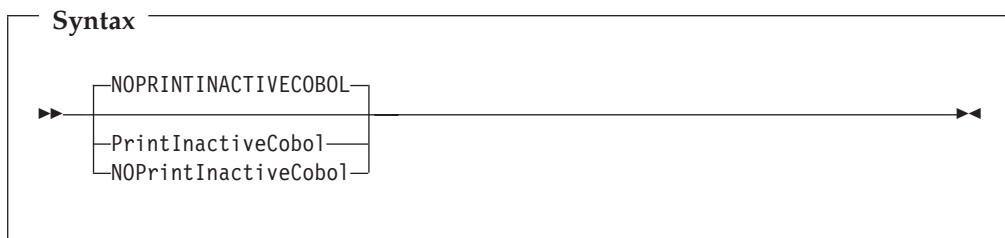
This option specifies the real-time or batch reanalysis report line width to be used for flowing paragraph text and hexadecimal storage formatting. Other parts of the report are not affected by this option.

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 from 80 to 132, both inclusive.

This option is not applicable to interactive reanalysis.

## PrintInactiveCOBOL



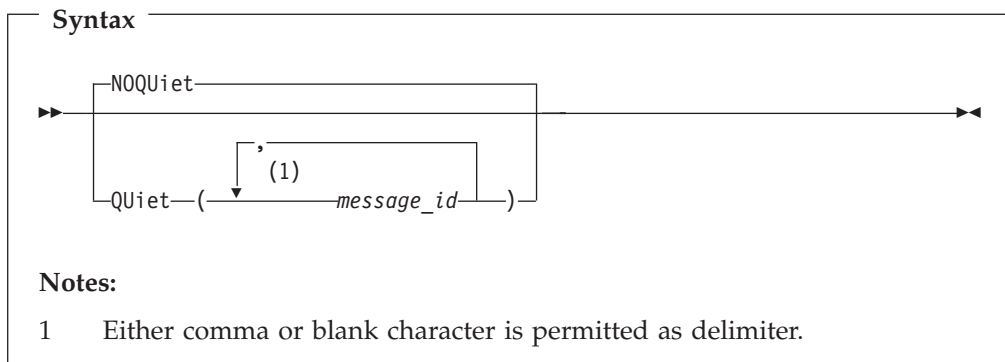
The PrintInactiveCOBOL option can be used with real-time or batch reanalysis to request that storage for inactive COBOL programs (programs that are not in the current save-area chain) is included in the analysis report.

**Note:** Depending on the number and size of inactive COBOL programs, this 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 will be 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.

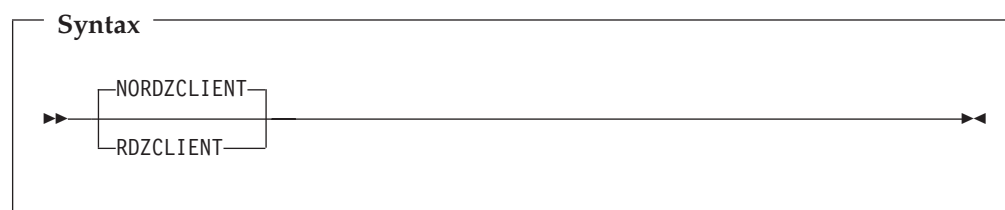
While this option can be used to reduce the number of messages written to the SYSLOG, its use is not generally recommended, since the lack of messages can make it difficult to diagnose problems with the Fault Analyzer processing. Also,

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

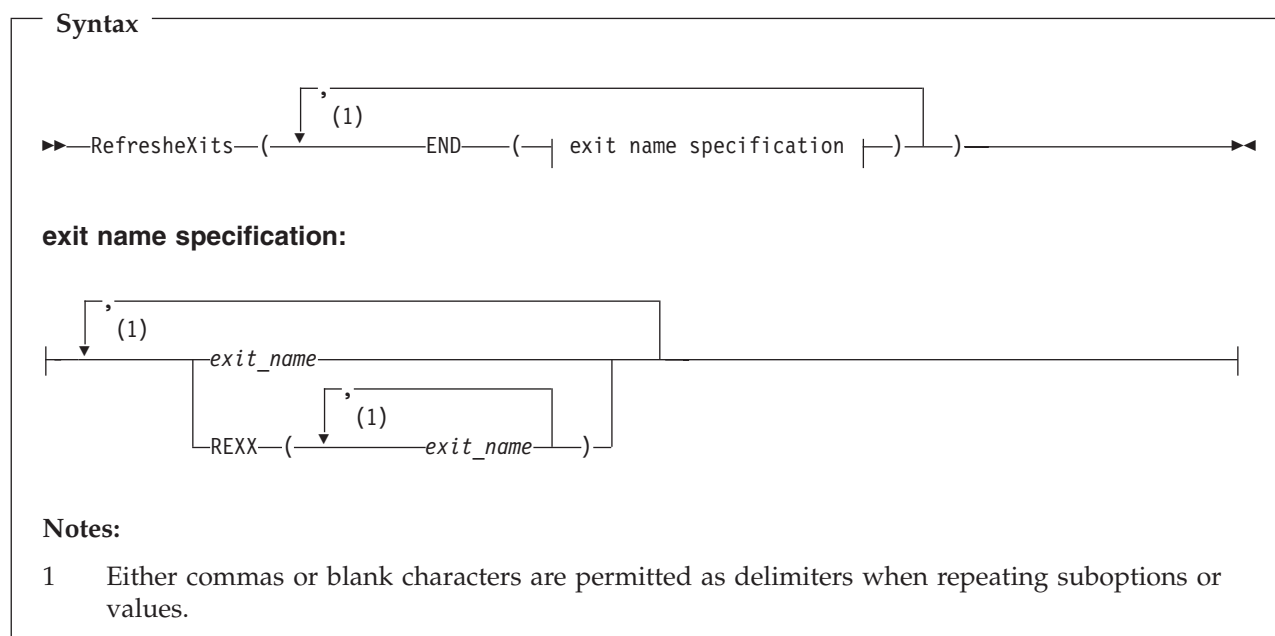


**Note:** The RDZClient option is equivalent to the earlier WZClient option, which is maintained for compatibility.

When the RDZClient option is in effect, then Fault Analyzer will write additional information about a fault to the history file fault entry. This information can improve the performance when viewing fault entries written with the RDZClient option in effect using an IBM Rational Developer for System z client.

This option is only included in the section of the fault analysis report that shows options in effect if real-time analysis with RDZClient in effect.

## 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 will be attempted invoked.

Multiple specifications of the RefreshExits option are cumulative.

Exits may 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 404.

The exit name specified as *exit\_name* may 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 will not be invoked and will cause 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 will be written to the “Options in Effect” section of the analysis report. In this section, all specified exits will be listed, and those of each type that were invoked (if any) will be identified.

An example of the information written to the “Options in Effect” section of the analysis report follows:

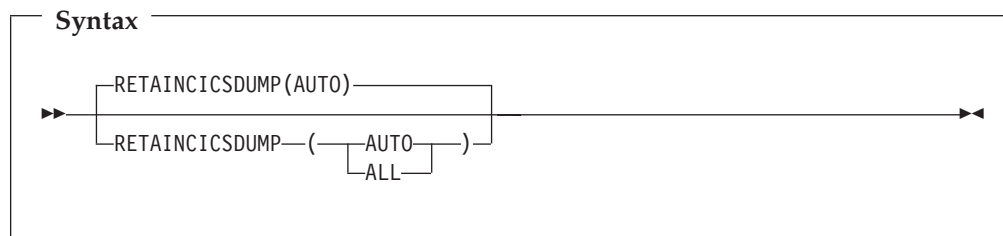
Exits:

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



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.

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 will not be suppressed by Fault Analyzer. 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.

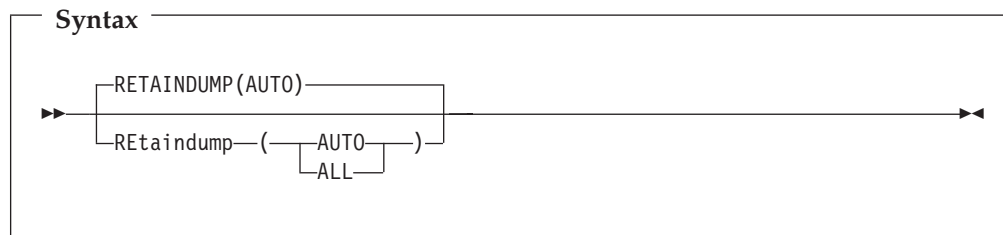
This option does not affect the writing of the fault entry to the history file.

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.

**Notes:**

1. The use of an End Processing user exit may 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 13 for additional 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,

## RetainDump

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.

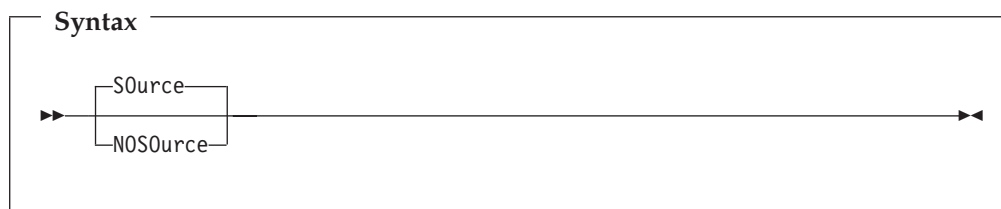
If an exit is installed ahead of IDIXDCAP in the IEAVTABX exit list, and this exit has requested, via its return code, that dump options should be changed or the dump suppressed, then Fault Analyzer honors the request regardless of options settings.

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.

### Notes:

1. The use of an Analysis Control or End Processing user exit may 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 13 for additional information about dump suppression.

## Source

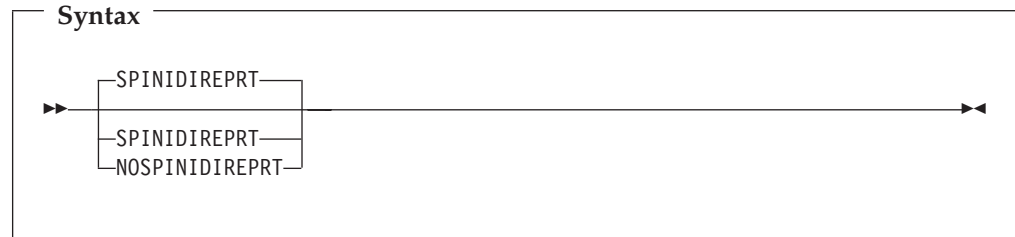


The Source option is used to control whether real-time source code analysis is to be performed:

- If Source is in effect, then real-time source code analysis will be performed as usual. This is the default.
- If NoSource is in effect, then Fault Analyzer will 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 additional source code information is an acceptable alternative.

## SpinIDIREPRT

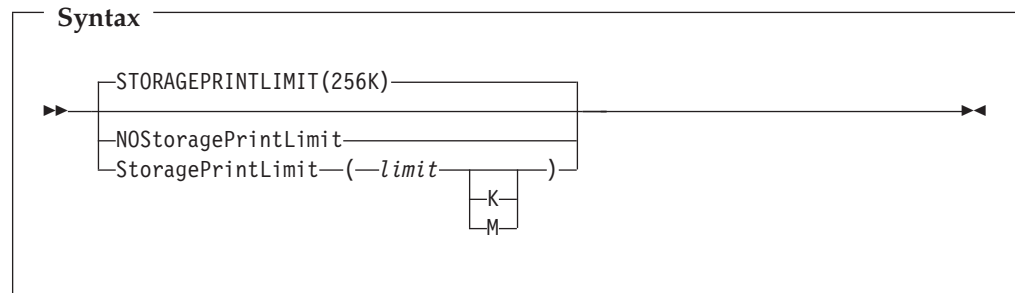


The NoSpinIDIREPRT option can be used with real-time analysis to request that dynamic allocation of the IDIREPRT DDname is not explicitly unallocated at the end of processing. This can, for example, be used to ensure that all JES spool data for a given real-time analysis is capable of being sent as a 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 unallocated at the end of processing.

This option is not applicable to batch or interactive reanalysis.

## 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 28. 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.

## StoragePrintLimit

To determine if the specified limit is exceeded, it is tested against the sum of all hex-dumped event-related associated 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 will still be 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)`.

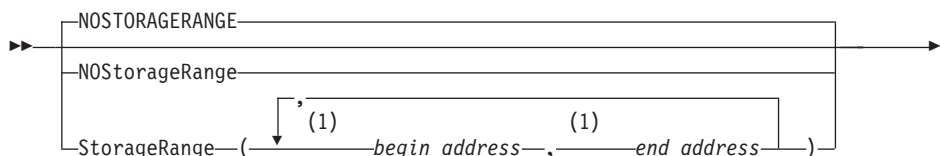
Information about the accumulated storage amount, which is used to determine whether or not the storage print limit in effect has been exceeded, is written to IDITRACE.

In the "Options in Effect" section of the report, this option will be 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 will be showed as `StoragePrintLimit(1M)`. If instead `StoragePrintLimit(1025)` was specified, then the option will be 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

### Syntax



### Notes:

- 1 Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

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 will instead contain 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 18.

If this option is in effect for real-time analysis, then the same specification will automatically be 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.

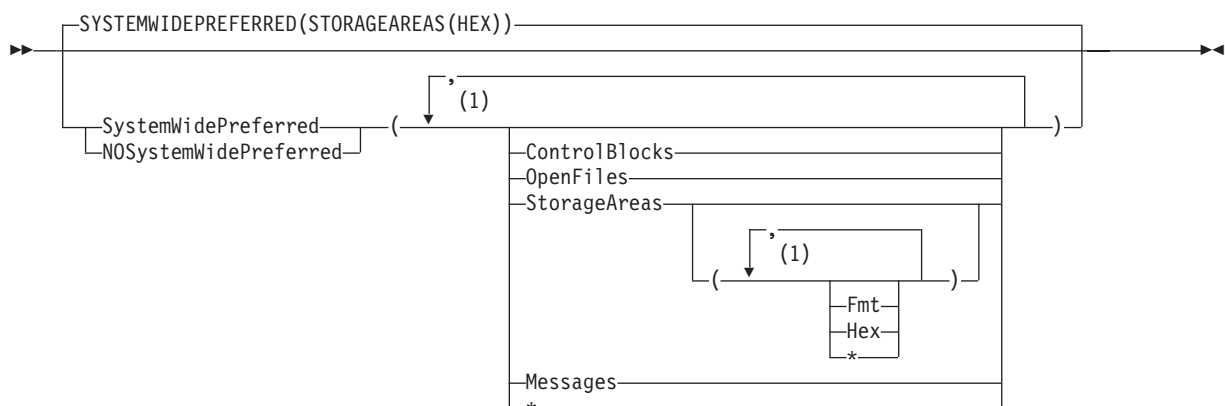
Multiple storage ranges can be specified in any order, as long as the end address is greater than the begin address. Any invalid storage ranges are ignored.

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

## 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.

## SystemWidePreferred

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 includes the following control blocks for the various execution environments:

*Table 29. Control blocks affected*

Execution environment	Control blocks affected	Target System-Wide Information section subheading
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 will be 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 labelling of addresses to indicate their origin. This 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

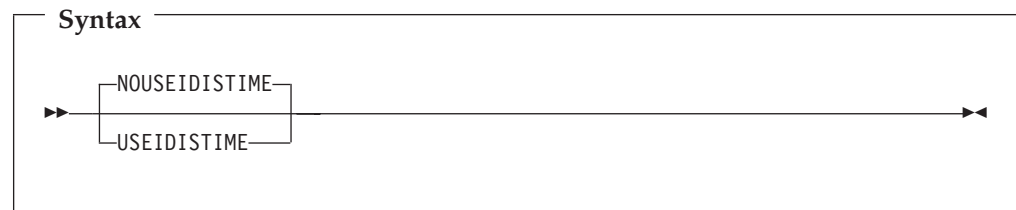
Specifies that all event-related messages that are usually placed under the heading "Associated Messages" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead.

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



When the UseIDISTime option is in effect, then Fault Analyzer will obtain all current date and time information from the IDIS subsystem. This might be useful if changing the system date to a future or past date for an application, since fault entry and report timestamps will be unaffected.

To enable this option, the IDIS subsystem must be started. Also, the IDIS subsystem must not be affected by the altered system time in order to achieve the intended result.

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 UseIDISTime in effect.



---

## 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 369.

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 additional information about tracing, see “Diagnostic tracing” on page 373. For additional information about the Exits option, see “Exits” on page 473.

---

### Non-REXX user exit buffered data format

The information 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 fields exceed the maximum field size, then Fault Analyzer will instead allocate 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 set list in the same blank-delimited format as when the data set names are 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 may not free any buffer allocated by Fault Analyzer for the field. Instead, if the field or buffer size is inadequate, the exit may 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 dependant on any original buffer address to be retained for later release of allocated storage.

Freeing of buffers allocated by user exits is the responsibility of the user. This can be achieved by 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 30. 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 will be performed, and no updates will be made to the history file.  Only applicable to real-time processing and dump registration.
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 495 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)

## CTL data area

Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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 503. The format of this field is transparent to users of REXX exits.  Only applicable to real-time processing.
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 503. 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 496 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 492 for information about this option.  Not applicable to interactive reanalysis.
325	(145)	CHAR	R/O	51	(Reserved)



Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
<b>Field format and usage:</b>					
<ul style="list-style-type: none"> <li>The following information is applicable to all IDI*_PRE, IDI*_JOB, and IDI*_CFG fields in this data area: 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 additional 25 data set names. For details, see “Non-REXX user exit buffered data format” on page 503. The format of this field is transparent to users of REXX exits.</li> <li>The following information is applicable to all IDI*_PRE fields in this data area: An Analysis Control user exit may choose to unallocate any of these data sets, or allocate additional ones. These fields contain the preallocated data set names, and are provided for information only. Changes to the data set names in this field will not be acted upon by Fault Analyzer. Instead, Fault Analyzer re-determines the preallocated data sets using standard operating system interfaces.</li> <li>The following information is applicable to all IDI*_JOB or IDI*_CFG fields in this data area: An Analysis Control user exit may 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 may be aligned on any boundary. For these data set names, Fault Analyzer will use the returned list of names.</li> </ul>					
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.
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 additional 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 additional 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 additional information.

## CTL data area

Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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 additional 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 additional 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 additional information.
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 additional 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 additional 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 additional information.

Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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 additional 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 additional 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 additional information.
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 additional 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 additional 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 additional information.

## CTL data area

Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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 additional 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 additional 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 additional information.
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 additional 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 additional 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.

Table 30. CTL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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 additional 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 additional 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 additional information.

## ENV - Common exit environment information

Table 31. ENV data area

Offsets		Type	Access	Len	Name and description
Dec	Hex				
0	(0)	CHAR	R/O	4	<b>VERSION</b>  Parameter list version (currently 0004).  Summary of changes:  <b>0003</b> Support for 6-digit CICS system abend codes.  <b>0004</b> 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: <b>C</b> Analysis Control <b>L</b> Compiler Listing Read <b>R</b> Batch Report Tailoring <b>M</b> Message and Abend Code Explanation <b>F</b> Formatting <b>E</b> End Processing <b>N</b> Notification <b>I</b> IDIUTIL Import <b>D</b> IDIUTIL Delete <b>H</b> IDIUTIL ListHF <b>X</b> Dump registration Analysis Control exit <b>Y</b> Dump registration Notification exit <b>Z</b> 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).
31	(1F)	CHAR	R/O	1	<b>REALTIME</b>  Real-time execution (Y/N).

Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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. (Not available to dump registration user exits.)
56	(38)	CHAR	R/O	8	<b>USER_ID</b>  User ID. (Not available to dump registration user exits.)
64	(40)	CHAR	R/O	4	(Reserved)
68	(44)	CHAR	R/O	8	<b>ABEND_MODULE_NAME</b>  ABEND module name. This 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 via the "File" 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)
86	(56)	CHAR	R/O	1	<b>JOB_CLASS</b>  Job execution class. (Not available to dump registration user exits.)
87	(57)	CHAR	R/O	3	<b>ACCOUNTING_FIELDS</b>  Number of job accounting fields (nnn) (from JCT ACTJNFLD). (Not available to dump registration user exits.)

## ENV data area

Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
90	(5A)	CHAR	R/O	144	<b>ACCOUNTING_INFO</b>  Job accounting information (from JCT ACTACCNT). (Not available to dump registration user exits.)  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.ACCOUNTING_FIELDS).  For step accounting, this field contains a three-byte maximum step running time, followed by a one-byte number of fields in step, followed by a one-byte length field, followed by the content of the field (the last two fields repeated as many times as the number of fields in byte 4 indicates) for each step.  Any non-printable characters (such as the binary field length value) are shown as periods.
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)
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 is equivalent to the LoopProtection option being in effect.</li> <li>• When set to N, this 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 will be ignored.  Refer to “LoopProtection” on page 480 for additional information about this option.  This option can be modified by any user exit.
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')



Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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: <b>C</b> CICS XPCABND exit (IDIXCX52 or IDIXCX53) <b>E</b> CICS LE CEEEXTAN exit (IDIXCCEE) <b>L</b> LE CEEEXTAN exit (IDIXCEE) <b>M</b> MVS IEAVTABX change options/suppress dump exit (IDIXDCAP) <b>S</b> Fault Analyzer program SNAP interface (IDISNAP) <b>P</b> 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. (Not available to dump registration user exits.)
265	(109)	CHAR	R/O	8	<b>JOB_ID</b>  JES job ID. (Not available to dump registration user exits.)
273	(111)	CHAR	R/O	8	<b>IMS_PROGRAM_NAME</b>  IMS program name. This is available if the environment has a DFSRPRX0 module loaded. (Not available to dump registration user exits.)
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 will no longer be running the abending transaction.

## ENV data area

Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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.
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 may choose to change this data set name to that of another history file, which will subsequently be used instead. For all other user exit types or processing modes, this field is read-only.  For a Notification user exit invoked for a duplicate fault (NFY.NFYTYPE='N' or 'F'), this is the name of the history file in which the duplicate fault identified by ENV.FAULT_ID was found.
417	(1A1)	CHAR	R/O	6	<b>ABEND_CODE</b>  The initial (or only) abend code: <ul style="list-style-type: none"> <li>• If ENV_JOB_TYPE = C, then this is a 4-character CICS transaction abend code.</li> <li>• Else if ENV_JOB_TYPE = I, then this 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>
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 234).
429	(1AD)	CHAR	R/O	9	<b>CICS_VRM</b>  CICS release level in VnnRnnMnn format (not available at the time of calling the Analysis Control user exit).

Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
438	(1B6)	CHAR	R/O	9	<b>DB2_VRM</b>  DB2 release level in VnnRnnMnn format (not available at the time of calling the Analysis Control user exit).
447	(1BF)	CHAR	R/O	9	<b>IMS_VRM</b>  IMS release level in VnnRnnMnn format (not available at the time of calling the Analysis Control user exit).
456	(1C8)	CHAR	R/O	9	<b>ZOS_VRM</b>  z/OS release level in VnnRnnMnn format (not available at the time of calling the Analysis Control user exit).
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 additional information about this flag, including specification of fault entry expiration control, see “Viewing fault entry information” on page 78.  By default, the lock flag is set to a blank, which will not prevent deletion of the fault entry.  This flag can be modified by any user exit. However, changes will only be 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.  Any printable character may be specified for this field: <ul style="list-style-type: none"> <li>• If a non-printable character is specified, then it will be changed to a '/'.</li> <li>• If a lowercase character is specified, then it will be translated to uppercase.</li> </ul>
467	(1D3)	CHAR	R/O	5	<b>DUPLICATE_COUNT</b>  Number of duplicate faults detected against the fault identified in ENV.FAULT_ID.
472	(1D8)	CHAR	R/O	8	<b>POF_MODULE_NAME</b>  Point-of-failure module name.
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.
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.

## ENV data area

Table 31. ENV data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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. (Not available to dump registration user exits.)
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 is a 4-character CICS transaction abend code.</li> <li>• Else if ENV_JOB_TYPE = I, then this 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, additional minidump pages resulting from storage that will be referenced during execution of a Formatting user exit are not included in this value.
646	(286)	CHAR	R/O	8	<b>IDIRLOAD_DD</b>  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. <b>Note:</b> Not applicable to real-time processing.
638	(27E)	CHAR	R/O	886	(Reserved)

## EPC - End Processing user exit parameter list

Table 32. EPC 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	4	(Reserved)
8	(8)	CHAR	R/W	1	<b>IS_DUPLICATE</b>  This 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 may 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 exceed 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 13.
121	(79)	CHAR	R/O	63	(Reserved)

## LST - Compiler Listing Read user exit parameter list

Table 33. 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).
4	(4)	CHAR	R/O	8	<b>MODULE_NAME</b>  Module name.  This 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 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: <ul style="list-style-type: none"> <li>• Assembler</li> <li>• C/C++</li> <li>• COBOL</li> <li>• OS/VS COBOL</li> <li>• PL/I</li> <li>• Entprs PL/I</li> </ul>
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.

Table 33. LST data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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.
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 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 382 for additional data set name requirements.

## NFY - Notification user exit parameter list

Table 34. 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 503. The format of this field is transparent to users of REXX exits.
1073	(431)	CHAR	R/O	1	<b>NFYTYPE</b>  C      Fault created R      Recovery fault recording N      NoDup(Normal) duplicate F      NoDup(CICSfast) or NoDup(ImageFast) duplicate
1074	(432)	CHAR	R/O	8	<b>DUPCOUNT</b>  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 35. UFM 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/W	100	<b>USEROPTIONTITLE</b>  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	<b>NUM_EVENTS</b>  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	<b>EVENT_NO</b>  Current event number (nnnnn).
205	(CD)	CHAR	R/O	5	<b>NEXT_EVENT_NO</b>  Next available event number (decimal).
210	(D2)	CHAR	R/O	5	<b>PREVIOUS_EVENT_NO</b>  Previous available event number (decimal).
215	(D7)	CHAR	R/O	1	<b>POF</b>  Point of failure (Y/N).

## UFM data area

Table 35. 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 503. The format of this field is transparent to users of REXX exits.  <div>Japanese feature note</div> If Language(JPN) is in effect, then the event type description provided in this field will be subject to translation into Japanese. <div>End of Japanese feature note</div> <div>Korean feature note</div> If Language(KOR) is in effect, then the event type description provided in this field will be subject to translation into Korean. <div>End of Korean feature note</div>
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 503. 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 503. 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 35. 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 503. 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 503. 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 503. 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 35. 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 35. 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 390. 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 35. 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 36. 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 37. 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.
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



Table 37. XPL data area (continued)

Offsets		Type	Access	Len	Name and description
Dec	Hex				
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.
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 386.
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)

## **XPL data area**

---

## 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	
---	--

- One or more informational messages might have been issued (message numbers suffixed by 'I').

2 or 4	
--------	--

- One or more warning messages has been issued (message numbers suffixed by 'W').
- Informational messages might also have been issued.

8	
---	--

- One or more error messages has been issued (message numbers suffixed by 'E').
- Informational and warning messages might also have been issued.

12	
----	--

- One or more severe messages has been issued (message numbers suffixed by 'S').
- Informational, warning and error messages might also have been issued.

---

### 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 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 will typically be presented in the format:

[*point-of-failure*:] *symptom*

where:

*point-of-failure*

Is 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*

Is 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 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 218 for additional information. If using TSO, specify a larger region size on the logon panel.

Fault Analyzer will deliberately attempt not to use up all available 31-bit storage before issuing this message, as this 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 identified in *reason*.

The context in which the data set was attempted opened is provided in *context*. This might be an associated DDname, or some other description of the type of data set involved.

When *reason* is in the form of

DYNALLO 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 may 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 may 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 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 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 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 may 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 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 will be created and no duplicate counts will be updated.

**User response:** Determine the reason for theabend.

---

#### 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 specified in *options-source*, where *options-source* is one of the following:

##### Analysis Control user exit

This indicates that the option was provided by an Analysis Control user exit.

##### Environment variable \_IDI\_OPTS

This indicates that the option was provided through the \_IDI\_OPTS environment variable.

##### Options line for interactive reanalysis

This indicates that 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

This indicates that the option was either specified in the PARM field of the EXEC card for PGM=IDIDA in a batch reanalysis job, or it was specified in the "Options line for batch reanalysis" field on the Batch Reanalysis Options display, which is shown when selecting "Batch Reanalysis Options..." from the action bar Options pull-down menu.

##### Parmlib config member

This indicates that the option was specified in the IDICNFxx parmlib member, or in a data set and member identified by a IDICNFUM user-options module.

##### User options file

This indicates that 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 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 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*.

**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 will terminate, 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 may 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 specified in *options-source*, where *options-source* is one of the following:

**PARM field**

This is the JCL EXEC statement PARM field specified in a job which executes program IDIDA.

**Environment variable \_IDI\_OPTS**

This environment variable will have been set by the abending application.

**System action:** Processing continues.

**User response:** Correct the error and resubmit your job.

---

**IDI0030W      Allocation to SYSOUT=\* 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 may 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 will 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 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 270).
- **Parmlib config member Exclude option specification**  
An Exclude option 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 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 will be 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 will be 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=*
//SYSDUMP 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**    **Fetch failed for message module**  
*module-name - using language*  
*old-language-option instead of*  
*new-language-option*

**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**    **Current fault is a duplicate of fault ID**  
*faultid - the duplicate count is count*

**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 will be suppressed.

**User response:** None

---

**IDI0046I**    **Dump registration skipped due to:**  
*reason*

**Explanation:** Dump registration was not performed for the reason identified in *reason* as one of the following:

- **Parmlib config member Exclude option specification**

An Exclude option 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 will be created.

**User response:** None

---

**IDI0047S**    **IBM Fault Analyzer internal abend**  
*abend-code*

**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 will be required for problem analysis. This 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

You might consider always having this SLIP trap enabled on your system.

---

**IDI0048W**    **Incomplete minidump: Expected**  
*expected-pages pages, read read-pages*  
**pages**

**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**    **Fault ID *faultid* not found in history file**

**Explanation:** A non-existing fault identifier was specified in the FaultID option.

**System action:** Processing terminates.

**User response:** Ensure that the history file used contains the fault ID specified in the FaultID option.

---

**IDI0050S**    **Fault reanalysis attempted without**  
**FaultID or DumpDSN option specified**

**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 will still be 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 will likely have been issued prior to this message.

**Duplicate fault or End Processing user exit**

If duplicate fault detection was the reason, then message IDI0044I will have been 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. This can be done by checking if earlier issued messages provide additional information, or by checking 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>27</sup> parameter. For additional information, see "Storage recommendations" on page 218.

**Note:** For CICS, this entire amount of storage must be provided through the JOB or EXEC JCL statement REGION<sup>27</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 page 28) will be 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 OS/390: TSO/E REXX Reference.

**System action:** Processing continues. However, no REXX exec user exits will be invoked and no diagnostic information will be written to the IDITRACE DDname.

**User response:** Refer to the description of return and reason codes in OS/390: TSO/E REXX Reference.

---

**IDI0057E**     Dynamic load of LE failed from IDICEEDS *module-name data-set-name*

**Explanation:** Either the LE run-time data set obtained from the IDICEEDS CSECT could not be opened (in which case no module name will be 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 run-time 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 run-time

---

<sup>27</sup> 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.

## IDI0058W • IDI0064W

library (++IDILED5)” on page 244 for information about the IDILED5 USERMOD that can be used to modify the default LE run-time data set name when LE is not in LINKLIST and the LE run-time data set name is not CEE.SCEERUN.

---

**IDI0058W**    **Allocation of temporary work data set failed: Type=type RC=return-code Error=error-code Info=info-code**

**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.

---

**IDI0059W**    **Logical parmlib concatenation list error: RC=return-code Reason=reason**

**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.

---

**IDI0061E**    **Data set organization of history file data-set-name is not PDS or PDSE**

**Explanation:** Fault Analyzer was invoked using an invalid format history file data set.

**System action:** Processing continues, but a fault entry can not be written.

**User response:** Ensure that the history file is either PDS or PDSE.

---

**IDI0062W**    **Error in file data-set-name. reason**

**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

might result in the loss of duplicate fault information.

---

**IDI0063W**    **Deletion of file data-set-name failed. reason**

**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.

---

**IDI0064W**    **CICS Exit command failed. Response=response reason=reason**

**Explanation:** A CICS command failed.

The following lists the possible response codes:

*Table 38. Response codes*

Response code	Meaning
1	OK
2	Exception
3	Disaster
4	Invalid
5	Kernel error
6	Purged

The following lists the reason codes associated with each command:

*Table 39. Reason codes*

Reason code	Meaning
<b>FREEMAIN</b>	
(No applicable reason codes)	
<b>GETMAIN</b>	
1	Insufficient storage
<b>INQ_APPLICATION_DATA</b>	
1	DPL program
2	No transaction environment
3	Transaction domain error
4	Invalid function
5	Abend
6	Loop
7	Inquiry failed
<b>INQUIRE_TRANSACTION</b>	
1	No transaction environment
3	Buffer too small
7	Invalid transaction token
37	Abend



Table 39. Reason codes (continued)

Reason code	Meaning
39	Loop
<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 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 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 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

concurrent CICS transaction abends, try increasing the CICS DSA size.

---

**IDIO076E      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.

---

**IDIO077I      Fault Analyzer internal diagnostic: text**


---

**Explanation:** This is an internal Fault Analyzer message used to display specific diagnostic information.

**System action:** Processing continues.

**User response:** None.

---

**IDIO078E      Open of fault history file *data-set-name* failed because: *reason***


---

**Explanation:** The history file identified by *data-set-name* could not be opened for the reason identified in *reason*.

If *reason* is shown as "DDfopenw() rc=X'800sssrc'", then this indicates an abend during open processing, where:

*sss*            Is the hexadecimal MVS system abend code.

*rc*             Is 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 260 for additional information.

---

**IDIO081I      IEWBIND unusual condition *function* *module-name* rc=*reason-code***


---

**Explanation:** An error occurred while using the Binder.

If *reason-code* is 83000526, then this indicates an unusual condition encountered for an input module. This 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(IDIO081I).

**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.

---

**IDIO082E      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 has been 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 OS/390 V5 Call Level Interface Guide and Reference*, *DB2 UDB for OS/390 V6 ODBC Guide and Reference*, or *DB2 UDB for OS/390 and z/OS V7 ODBC Guide and Reference*, depending on the version of DB2 used).

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 233.

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 Universal Database for z/OS Version 8: Codes*

---

**IDIO083E      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.

---

**ID10084E      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.

---

**ID10085E      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.

---

**ID10086E      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 additional information, see “Storage recommendations” on page 218.

Although Fault Analyzer was unable to perform real-time analysis due to storage constraints, recovery fault recording (see page 28) will be attempted in a non-CICS environment.

---

**ID10087I      *size* Meg of 31 bit storage could be provided by SETPROG LPA,ADD,MOD(*module-list*),DSN=LNLST**


---

**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 additional information, see “Storage recommendations” on page 218.

Message ID10055E will precede 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 ID10055E 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.

---

**ID10088E      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 *xxxyyy*, 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.

---

**ID10089I      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 Universal Database for z/OS Version 8: Codes*.

**System action:** Processing continues, but the analysis might be incomplete.

**User response:** See if another message, such as ID10082E, was also issued, which might help explain the situation.

---

**ID10090E      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 the OS/390 or z/OS TSO/E 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 will be required for problem analysis. This 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

You might consider always having this SLIP trap enabled on your system.

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 480.

---

**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 is an environment which is not supported by REXX, then no REXX services will be available during the fault analysis. This 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 5 on page 225) 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 5 on page 225) 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 5 on page 225) 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 will be required for problem analysis. This 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
```

You might consider always having this SLIP trap enabled on your system.

---

**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 will not be 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 might be due to 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 will show 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 which cannot be cached in the IDIS subsystem.

For more information about IDIS subsystem caching, see "Caching of history file \$\$INDEX data" on page 234.

**System action:** Processing continues, but all I/O to the history file \$\$INDEX member will be performed by the requestor (real-time analysis, batch or interactive reanalysis, or Fault Analyzer ISPF interface).

**User response:** If you wish to utilize 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 additional 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 will be 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 333.

---

**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 might either be due to incomplete setup of the interactive reanalysis component under CICS (see "Performing interactive reanalysis under CICS" on page 194), 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, requestor 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 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 an additional 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 will need 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 will be performed, but the duplicate count of the earlier occurring fault will be incremented, and the duplicate

## IDI0122W • IDI0125W

details recorded for later viewing with the Fault Analyzer ISPF interface (see “Viewing the fault entry duplicate history” on page 83).

**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 will 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 IEFIC 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** *tcid-address*

Fault Analyzer was invoked to analyze an abend with the SYSZTIOT major name already enqueued by the TCB identified by the address in *tcid-address*. An example of when this might occur is during shutdown of IMS when Fault Analyzer is invoked for a U0002 abend.

- **Executing protect KEY 2 (WAS CR)**

Fault Analyzer was invoked to analyze an abend in a WebSphere Control Region.

**System action:** Processing terminates.

**User response:** Although real-time analysis is not possible under these conditions, Fault Analyzer might still be used to aid with the problem determination by first setting a SLIP trap using the MSGID=IDI0123S and A=SVCD parameters, and then performing analysis of the resulting SVC dump via 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 will be automatically restarted as necessary.

Requestor 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 will be automatically restarted as necessary.

Requestor 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 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 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. This 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 additional information 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 additional information, see “Recovery fault recording” on page 28.

---

**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 will continue 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.

**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 229).

---

**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 additional 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 will be 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 requestor 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 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 will be re-issued at intervals while the storage usage remains high, but will cease to be 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 IDI\_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 229 for information about how to

## IDI0149W • IDI0155W

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 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 specified for the IDIMAPS DDname is correct (this 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 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 on page 551 for additional 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 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** Installation options module IDIOPTLM found in non-authorized load library and has been ignored

**Explanation:** Load module IDIOPTLM, which can be used to provide installation options for Fault Analyzer, was found in a load library which was not APF-authorized. 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 IDIOPTLM 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 will instead use 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 occurred because of storage shortage in the requestor address space identified by *jobname* and *job-id*.

**System action:** Processing continues.

**User response:** If possible, increase the region size of the requestor 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 additional information about the use of the IDIRFRON DDname, see “Verifying the recovery fault recording set-up” on page 351.

**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 will be missing.

**User response:** Add IDI.SIDIAUT2 to the IDIS subsystem STEPLIB concatenation. For details, see “Starting the IDIS subsystem” on page 235.

---

**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 re-writing 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 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 will only be implicitly deleted if the history file contains more than 25 fault entries.

- **IGWFAMS error**

Message IDI0095W will be 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 might be due to fault entries being locked.

**System action:** Processing continues, but a fault entry will not be written to the history file.

**User response:** One or more of the following might be appropriate:

- Reallocate the history file with additional space.
- Change the history file space management setting using the IDIUTIL SetMaxFaultEntries 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.

**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
- IDIRegisterFaultEntry command

**System action:** Processing continues.

**User response:** None.

---

**IDI0164I      Fault ID *fault-id* created in history file *history-file* due to *reason***

---

## 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*.

---

## 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/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.debugt/debugt/6.1z/TrainingEducation/SupportInfoADTools/player.html>.

- “Searching knowledge bases”
- “Getting fixes” on page 559
- “Subscribing to support updates” on page 559
- “Contacting IBM Support” on page 560

---

### 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](http://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](http://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](http://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 additional 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 additional 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. Additional information is provided at <http://www.ibm.com/support/docview.wss?rs=615&uid=swg21172598>.

1. Go to the IBM software support site at <http://www.ibm.com/software/support>.

2. Click the **My Notifications** link in the **Notifications** portlet on the page that is displayed.
3. If you have already registered for **My notifications**, sign in and skip to the next step. If you have not registered, click **register now**. Complete the registration form using your e-mail address as your IBM ID and click **Submit**.
4. In the **My notifications** tool, click the **Subscribe** tab to specify products for which you want to receive e-mail updates.
5. To specify Problem Determination Tools products, click **Other software** and then select the products for which you want to receive e-mail updates, for example, **Debug Tool for z/OS** and **File Manager for z/OS**.
6. To specify a COBOL or PL/I compiler, click **Rational** and then select the products for which you want to receive e-mail updates, for example, **Enterprise COBOL for z/OS**.
7. After selecting all products that are of interest to you, scroll to the bottom of the list and click **Continue**.
8. Determine how you want to save your subscription. You can use the default subscription name or create your own by entering a new name in the **Name** field. It is recommended that you create your own unique subscription name using something easily recognized by you. You can create a new folder by entering a folder name in the **New** field or select an existing folder from the pulldown list. A folder is a container for multiple subscriptions.
9. Specify the types of documents you want and the e-mail notification frequency.
10. Scroll to the bottom of the page and click **Submit**.

To view your current subscriptions and subscription folders, click **My subscriptions**.

If you experience problems with the **My notifications** feature, click the **Feedback** link in the left navigation panel and follow the instructions provided.

---

## 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/offerings.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" on page 562
3. "Submit the problem to IBM Support" on page 562

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 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 encountered with Debug Tool for z/OS: [http://www.ibm.com/support/docview.wss?rs=615&context=SSGTSD&q1=mustgather&uid=swg21254711&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=615&context=SSGTSD&q1=mustgather&uid=swg21254711&loc=en_US&cs=utf-8&lang=en)
- Mustgather: Read first for problems 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 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 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 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 re-create the problem? If so, what steps were performed to re-create 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 has been 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 may be useful, for example to MATCH on today's date, or a specific PROGRAM name.

An example of how this 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 169 on page 566). A '?' can be placed in the Application/View field to display a list of available applications and views (see Figure 170 on page 566). 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 will also be 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 which is concatenated to the DDname indicated in the table.

*Table 40. Sample files*

File	DDname	Description
IDISFEMA	SYSPROC	Main REXX exec
IDISFESK	ISPSLIB	ISPF skeleton for creating history file
IDISFECL	SYSPROC	Intermediary CLIST used when invoking Fault Analyzer
IDISFEAP	ISPLIB	ISPF panel used for application selection
IDISFEQP	ISPLIB	Query ISPF panel
IDISFEMP	ISPLIB	Main ISPF panel for supplying user parameters

## Sample customized ISPF interface front-end

Menu Utilities Compilers Options Status Help		
Opti	Fault Analyzer History File Selection -----	
0 S		K
1 V	DSN: ADRIAN	
2		
3 U	Environment: FAE1	
4 F		H
5 B	Program : IDIXFA	
6 C		
7 D	Application: ADRIAN Enter ? for list	K
9 I	or Views	
10 S		
11 W	Enter=Check For DSN PF3=Exit	.0
12 z		
13 z		
14 I		
S SDSF	SDSF	

Figure 169. Sample display 1

Menu Utilities Compilers Options Status Help		
Opti	Fault Analyzer History File Selection -----	
0 S		K
1 V	DSN: ADRIAN	
2		
3 U	Envi	Row 1 to 10 of 10
4 F	Command ==>	H
5 B	Prog	Application Selection -----
6 C		
7 D	Appl	
9 I	or V	K
10 S		
11 W	Ente	.0
12 z		
13 z		
14 I		
S SDSF		
Enter		
Please use S to select the application. Application - AA Application 1 - AB Application 2 - AC Application 3 - ZZ Application 4 - FA Fault Analyzer Default - Dev1 View 1 - Dev1 View 2 - APC View 2 - DB2 View 3 - CICS View 4 ***** Bottom of data *****		

Figure 170. Sample display 2

Once a Fault History File or View name has been selected and verified, the sample code will perform 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 will be 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 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) will disappear.

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 will not be 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 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 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 will be 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)'
```

---

## 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 wish 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  
555 Bailey Avenue

## Notices

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 for the purpose of collecting 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

**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 SYSDUMP 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 may 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 a 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.

## Glossary

### 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 368
- \_IDI\_OPTS environment variable
  - options 455
- \_IDI\_OPTSFILE environment variable
  - options 454
- DROPCNF-
  - data set name 460
  - user exit name 475
- HistCols 252
- Match 252
- \$\$BACKUP history file data set
  - member 9
- \$\$INDEX history file data set member 9
- \$\$UFMTX example 155
- &SYSCONE MVS system symbol
  - used as IDICNFxx parmlib member suffix 267

## A

- ABCODE
  - using with EXEC CICS ABEND command 322
- ABCODE keyword on EXEC CICS ABEND command 322
- abend job information 129, 167
- abend job information display
  - example 129, 168
- about Fault Analyzer
  - help menu option 60
- about Fault Analyzer display
  - example 76
- action-bar pull-down menus 58
- ADATA option
  - used to produce SYSADATA file 303
- add blank lines
  - view menu option 60
- add help text
  - view menu option 60
- add pseudo assembler instructions
  - view menu option 61
- adding or removing blank lines 72
- adding or removing help text 72
- adding the required programs to the shutdown PLT 321
- adding the required programs to the startup PLT 320
- additional region size required 11
- AdditionalIDIOffDD option 455
- ADDR tag 436
- address TSO REXX commands 372
- allocate ISPF data sets 239
- allocating a PDS or PDSE for a history file 249
- allocating ISPF data sets for the Japanese feature 337
- allocating ISPF data sets for the Korean feature 339
- Analysis Control user exit 377

- Analysis Control user exit (dump registration) 380
- analysis report
  - See also* reanalyzing
  - real-time 16
- analyze MVS dump data set
  - file menu option 59
- analyze MVS dump data set display
  - example 164
- analyzing in real time 13
- application error handling
  - effect on invocation of Fault Analyzer 224
- application-handled error conditions 224
- applying an action to a particular fault 49
- AREA tag 436
- assembler ADATA option 303
- assembler exits
  - CICS NoDup(CICSFAST)
    - override 326
- assembling programs for IBM Problem Determination Tools 300
- associated file control blocks display
  - example 112
- associated storage areas display
  - example 131
- associated storage areas display with hex value column collapsed example 132
- associated storage areas display with level 88 items collapsed example 133
- authorizing Fault Analyzer 225
- AUTO-managed PDSE history files 250
- available dump status 82

## B

- batch dump reanalysis
  - submitting 50
- batch options line 90
- batch reanalysis 7
  - data sets used 94
  - initiating 94
  - JCL control statements 242
  - options 89
  - purpose 89
- batch reanalysis options
  - options menu option 60
- batch reanalysis options display
  - example 90, 92
- batch utility
  - IDIUTIL 353
- binder-related dependencies 10
- binding DB2 331
- blank lines
  - adding or removing 72

## C

- CA-Panexec
  - exit for 247
- caching of \$\$INDEX data 234
- call depth
  - maximum 187
- calling a non-REXX logging routine from REXX 407
- CEEWUCHA
  - special processing 11
- CEEWUCHA LE user condition handler 183
- CFA 323
- CFA Exit Options display example 324
- CFA IVP testing display example 346
- CFA transaction display example 323
- change fault history file settings
  - file menu option 59
- change fault history file settings display
  - example 55
- changing headings 195
- changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit 247
- changing the history file or view displayed 35
- changing the NoDup(CICSFAST)
  - option 488
- changing the NoDup(ImageFast)
  - option 486
- checklist for installing and customizing Fault Analyzer 213
- CICS
  - controlling transaction abend analysis 323
  - criteria for duplicate fault determination 487
  - customizing for 319
  - ensuring transaction abend analysis is not suppressed by DUMP(NO) 326
  - IVP testing 345
  - storage requirements 329
  - using IDIOPTS DDname 328
- CICS auxiliary trace data set formatting
  - selecting trace data set 173
- CICS definitions
  - sample job 322
- CICS Explorer plug-in
  - customize the IBM Problem Determination Tools for z/OS Common Component Common Server 445
  - download and install CICS Explorer 446
  - download Fault Analyzer
    - plug-in 446
  - install Fault Analyzer plug-in into CICS Explorer 446
- CICS fast duplicate fault suppression 403, 488

- CICS information
  - system-wide information 112
- CICS information display example 112
- CICS levels, commareas, and channels
  - system-wide information 117
- CICS levels, commareas, and channels display example 118
- CICS NoDup(CICSFAST) override
  - assembler exit (IDINDFUE) 326
- CICS open (L9) TCB
  - capturing abends on 329
- CICS region SDSF job data set display example 325
- CICS system abend analysis
  - creating history file entry 171
  - displaying the interactive report 166
  - selecting dump data set 163
  - setting options 163
  - user exit usage 163
- CICS system abend dump analysis 163
- CICS system abend interactive reanalysis report display example 166
- CICS system abend synopsis display example 167
- CICS system information 168
- CICS system information display example 169
- CICS trace display example 117
- CICS trace formatting
  - system-wide information 115
- CICS trace selection parameters display example 116
- CICS trace, formatting
  - system-wide information 115
- CICS trace, summarized
  - system-wide information 114
- CICS tracing 322
- CICS transaction abend analysis
  - performance
    - maximizing 329
- CICSDumpTableExclude option 456
- clear last accessed information
  - file menu option 59
- clearing the NoDup(CICSFAST(...))
  - recording area 324
- COBOL for MVS and VM
  - compiling programs for 280
- COBOL Report Writer Precompiler 307
- COBOL SCLM example 307
- COBOL suppressed copybooks 185
- COBOL SYSDEBUG file usage 309, 311
- collapsing level 88 items 132
- COLS command 61
- column configuration
  - view menu option 61
- column configuration source
  - identification 42
- columns available for fault entry list display 42
- com.ibm.faultanalyzer.Snap.Dump
  - method
    - using to invoke Fault Analyzer from Java 22
- combining Fault Analyzer real-time reports 17
- commands
  - COLS 61
- commands (*continued*)
  - COPY 62
  - DISASM 62
  - DSECT 62
  - DUPS 63
  - EXEC 63
  - FIND 63
  - INFO 65
  - ISPF interface 61
  - LOOKUP 66
  - MATCH 66
  - NEXT 67
  - NOTELIST 67
  - PREV 67
  - QUIT 68
  - reading syntax diagrams xi
  - REFRESH 68
  - RESET 68
  - RPTFIND 68
  - RUNCHAIN 69
  - SHOW 69
  - STCK 70
  - VIEWS 70
- comments
  - in options 452
- compiler listing display example 136, 137, 138
- compiler listing not found display example 157
- compiler listing or side file
  - prompting for 156
- compiler listing or side file selection criteria 11
- compiler listing or side files
  - using temporary data sets 314
- Compiler Listing Read user exit 382
- compiler listings
  - attributes 313
  - example of providing for reanalysis 95
  - naming 310
- compiler listings and side files 303
- compiler options for IBM Problem Determination Tools 273
- compiler options required for IDILANGX 308
- compiling a program and storing a side file 304
- Compiling for PD Tools 273
- compiling programs
  - COBOL for MVS and VM 280
    - Enterprise COBOL for z/OS Version 3 278
    - Enterprise COBOL for z/OS Version 4 275
    - Enterprise PL/I for z/OS Version 3.5 and earlier 291
    - Enterprise PL/I for z/OS Version 3.5 and Version 3.6 288
    - Enterprise PL/I for z/OS Version 3.7 and later 285
  - OS/VS COBOL 284
  - PL/I for MVS and VM 294
  - VS COBOL II 282
  - z/OS XL C and C++ 296
- Compiling programs for IBM Problem Determination Tools 273
- concurrent use of LE abnormal termination exit and MVS change options/suppress dump exit 221
- configuration-options module
  - IDIOPTLM 454
- configuring Language Environment for CICS to invoke Fault Analyzer 320
- confirm exit display example 104
- confirm fault entry deletion display example 78
- confirm fault entry deletion option
  - changing 77, 78
  - deleting when not set 78
  - deleting when set 77
- Confirm Java Fault Entry Reanalysis display example 177
- confirm sysmdump open display example 99
- controlling CICS transaction abend analysis 323
- controlling prompting 160
- controlling real-time analysis with options 15
- controlling report detail 270
- controlling the SYSOUT class of real-time reports 17
- controlling which jobs are analyzed with exclude processing 269
- converting STORE CLOCK values (STCK command) 152
- COPY command 62
- copy current display to data set
  - services menu option 60
- copying a history file entry 50, 86
- copying a range of fault history entries 86
- copying interactive displays to a file 75
- copyright and general usage information displaying 76
- CPU time used by Fault Analyzer 416
- create history file entry display example 172
- Create Java Fault Entry display example 176
- creating a history file entry 171
- creating a history file fault entry 175
- creating a side file from a listing 304
- creating and managing user notes 140
- creating side files using IDILANGX 303
- criteria for compiler listing or side file selection 11
- criteria for successful analysis 519
- cross-system coupling facility 261
- Cross-system Coupling Facility (XCF) 234
- CTL data area 505
- cultural environment
  - specifying 271
- cursor match 49
- cursor-selecting a matching value 46
- customer support
  - See Software Support
- customize Fault Analyzer by using USERMODs 243
- customizing Fault Analyzer by using user exits 369
- customizing for ISPF 239

- customizing the CICS environment 319
- customizing the DB2 environment 331
- customizing the Fault Analyzer Japanese feature 337
- customizing the Fault Analyzer Korean feature 339
- customizing the fault entry list display 271
- customizing the IMS environment 335
- customizing the operating environment for Fault Analyzer 225

## D

- data set name substitution symbols 461
- data set security server access requirement
  - IDIDOC 227
  - IDIVSENU 227
  - IDIVSJPN 228
  - IDIVSKOR 228
- data sets used for batch reanalysis 94
- data sets used for interactive reanalysis 160
- data sets used for reanalysis 460
- DATA tag 437
- DataSets option 458
- DB2
  - customization 331
  - IDIS subsystem requirements 237
  - Language Environment considerations 331
  - stored procedures 331
- DB2 and Language Environment 331
- DB2 information
  - system-wide information 120
- DB2 information display example 123
- DB2 IVP
  - using C 347
  - using COBOL 349
- DB2 stored procedures 331
- DD tag 437
- DDnames
  - IDIRLOAD 314
- DEBUG option considerations 310
- default options
  - installation-wide 267
- DeferredReport option 463
- defining program control access to Fault Analyzer programs 226
- defining required programs to CICS 320
- deinstalling Fault Analyzer temporarily 367
- delete confirmation display
  - deleting when not shown 78
  - deleting when shown 77
- DELETE control statement 355
- deleting a history file entry 76
- deleting a range of fault history entries 76
- deleting an history file entry 50
- deleting many fault history entries 78
- deletion options
  - setting from the Options menu 77
- descriptor codes
  - used by Fault Analyzer WTO messages 12

- Detail option 465
- detailed event information 107
- determining what size history files to allocate 249
- DFHRPL 322
- directing CICS transaction LE dump output to the CICS dump data set 328
- disabling Fault Analyzer 367
- DISASM command 62
- disassembling object code (DISASM command) 150
- displaying associated storage areas 130
- displaying chained data areas (RUNCHAIN command) 148
- displaying Fault Analyzer copyright and general usage information 76
- displaying source code 135
- displaying storage locations 138
- displaying the CICS system abend interactive report 166
- displaying the Java information in the interactive report 177
- displaying user-selected message or abend code explanations 73
- displays
  - abend job information 129, 168
  - about Fault Analyzer 76
  - analyze MVS dump data set 164
  - associated file control blocks 112
  - associated storage areas 131
  - associated storage areas with hex value column collapsed 132
  - associated storage areas with level 88 items collapsed 133
  - batch reanalysis options 90, 92
  - CFA IVP testing 346
  - CFA transaction 323, 324
  - change fault history file settings 55
  - CICS information 112
  - CICS levels, commareas, and channels 118
  - CICS region SDSF job data set 325
  - CICS system abend interactive reanalysis report 166
  - CICS system abend synopsis 167
  - CICS system information 169
  - CICS trace 117
  - CICS trace selection parameters 116
  - compiler listing 136, 137, 138
  - compiler listing not found 157
  - confirm exit 104
  - confirm fault entry deletion 78
  - Confirm Java Fault Entry Reanalysis 177
  - confirm sysmdump open 99
  - create history file entry 172
  - Create Java Fault Entry 176
  - DB2 information 123
  - dump storage 138
  - event details 109
  - event summary 106
  - Fault Analyzer options 130
  - Fault Analyzer preferences 77
  - fault entry duplicate history 84
  - fault entry information 79
  - fault entry list 33, 34

- displays (*continued*)
  - fault entry list column configuration 41
  - fault entry list with updates pending 57
  - File Browse 40
  - file information 111
  - format CICS auxiliary trace data set 173
  - Formatting User Exit Selection List 155
  - hex-dumped storage 127
  - history file properties 51
  - history file updates pending 58
  - IMS information 125
  - interactive reanalysis options 97
  - interactive reanalysis report 103
  - interactive reanalysis status 102
  - Java event details 180
  - Java event summary 179
  - Java information 182
  - Java interactive reanalysis report 178
  - Language Environment heap analysis information 128
  - last accessed history file entries 39
  - last accessed history files or views 38
  - last CICS 3270 screen buffer 113
  - last CICS 3270 screen buffer hex 114
  - level 88 items 134
  - listing/side file mismatch 159
  - listing/side file trace 160
  - lookup search and browse 74
  - message explanation 135
  - message id look-up 74
  - MTRACE records 128
  - new history file allocation 53
  - options in effect 171
  - preferred formatting width 73
  - real-time report 71
  - specify compiler listing or side file 158
  - specify move/copy options 86
  - specify XMIT options 88
  - STCK conversion 153
  - storage disassemble 151
  - storage DSECT mapping entry 146
  - storage DSECT mapping map 147
  - storage RUNCHAIN command entry 149
  - summarized CICS trace 115
  - synopsis 105
  - system-wide messages 120
  - system-wide open files 110
  - system-wide storage areas 126
  - user fields update prompt 83
  - user note list 144
  - user notes update prompt 145
  - view list 40
- DL tag 437
- dropping IDICNF00 data sets 460
- dropping IDICNF00 user exits 475
- DSECT command 62
- DSECT indexing utility (IDIPDSU) 148
- DSECT information
  - mapping of storage areas using 145
- DSNACLI default DB2 plan 331
- DT tag 438



- DUMMY data set
  - specifying via DataSets option 460
- dump data set name
  - viewing 50
- dump data set size 224
- dump registration
  - Analysis Control user exit 380
  - Fault Analyzer IDIS subsystem usage 233
  - identification of fault entry 43
  - IDIXTSEL invocation exit 222
  - indication of invocation exit in ENV data area 515
  - Notification user exit 411
  - process description 24
  - specifying user exits for 466
- dump status
  - available 82
  - not found 82
  - suppressed 82
- dump storage display example 138
- dump suppression 13
  - controlling with user exit 403
- DUMP tag 439
- DUMP(NO) 326
- DUMPA tag 439
- DumpDSN option 466
- DumpRegistrationExits option 466
- duplicate fault count 81
- duplicate fault detection
  - overview 482
- duplicate fault determination
  - controlling with user exit 402
  - criteria 489
- duplicate fault processing
  - overview 28
- duplicate history
  - showing 50
- DUPS command 63
- dynamic SVC update 225

## E

- eliminating the need for a dump DD statement 245
- enabling dynamic control of analysis of CICS transaction abends 321
- enabling Fault Analyzer to be invoked 243
- enabling implicit Fault Analyzer invocation from PL/I V2R3 applications 245
- enabling the Language Environment
  - abnormal termination exit IDIXCEE 243
- End Processing user exit 402
- End Processing user exit (fault entry refresh) 404
- ensuring transaction abend analysis is not suppressed by DUMP(NO) 326
- Enterprise COBOL for z/OS Version 3
  - compiling programs for 278
- Enterprise COBOL for z/OS Version 4
  - compiling programs for 275
- Enterprise PL/I for z/OS Version 3.4 and earlier
  - compiling programs for 291

- Enterprise PL/I for z/OS Version 3.5 and Version 3.6
  - compiling programs for 288
- Enterprise PL/I for z/OS Version 3.7 and later
  - compiling programs for 285
- Enterprise PL/I SYSDEBUG file usage 309, 311
- ENV data area 512
- environment variable \_IDL\_OFF 368
- EPC data area 519
- EQALANGX 306
- EQAUEDAT exit
  - locating SYSDEBUG files 311
- error display example 34
- error handling in applications
  - effect on invocation of Fault Analyzer 224
- ErrorHandler | NoErrorHandler option 467
- Evaluate REXX command 418
- event details display example 109
- event summary 105
- event summary display example 106
- Exclude option 468
- EXEC command 63
- exit Fault Analyzer
  - file menu option 59
- Exit from the interactive report 104
- exit interactive reanalysis
  - file menu option 59
- exits
  - CICS 221
  - invocation 219
  - Language Environment 220, 222
  - MVS 219, 222
  - user 369
- EXITs control statement 359
- exits for invoking Fault Analyzer 219
- Exits option 473
- expanding messages and abend codes 134

## F

- FA exec 240
- FA line command for ISPF 3.4 data set list 240
- FA standard date format 510
- fast Exclude options processing 270
- fastpath navigation for CICS system
  - dump analysis 167
- fault analysis report
  - viewing 50
- Fault Analyzer and CICS global user exits 14
- Fault Analyzer client for IBM Rational Developer for System z 446
- Fault Analyzer ISPF interface 31
- Fault Analyzer options 129
- Fault Analyzer options display example 130
- Fault Analyzer plug-in for Eclipse 193, 445
- Fault Analyzer preferences
  - options menu option 60

- Fault Analyzer preferences display example 77
- fault entries
  - finding 46
  - matching 46
  - selecting 46
- fault entry duplicate history
  - viewing 83
- fault entry duplicate history display example 84
- fault entry expiration control 58
- fault entry information
  - file menu option 59
  - viewing 78
- fault entry information display example 79
- fault entry list column configuration display example 41
- fault entry list display
  - customizing 271
- fault entry list display example 33
- fault entry list display with updates
  - pending example 57
- fault entry refresh
  - End Processing user exit 404
- fault entry refresh processing 161
- fault history entries
  - copying 86
  - cursor match 49
  - deleting 76
  - deleting many 78
  - moving 87
  - transmitting 87
- fault identifier
  - format 475
- fault reanalysis 6
  - batch 89
  - creating your own job 95
  - interactive 97
- FaultID option 475
- File Browse display example 40
- file information display example 111
- File Manager
  - ISPF data set allocations required 239
- FILES control statement 354
- FIND command 63
- FIND command differences between display types 65
- finding fault entries 46
- fixes, getting 559
- FND area
  - clearing 324
- format CICS auxiliary trace data set
  - file menu option 60
- Format CICS Auxiliary Trace Data Set display example 173
- formatting a CICS auxiliary trace data set 173
- formatting tags
  - ADDR 436
  - AREA 436
  - DATA 437
  - DD 437
  - DL 437
  - DT 438
  - DUMP 439



formatting tags (*continued*)

- DUMPA 439
- HP 440
- L 440
- LI 441
- NOTEL 441
- P 441
- TH 442
- U 442
- UL 443

Formatting user exit 390

Formatting User Exit Selection List  
example 155

fragments, syntax diagrams xi

## G

general information about the interactive  
report 102

general report information 183

GenerateSavedReport option 476

global environment data area (ENV) 372

global resource serialization 261

GRS 261

## H

help text

adding or removing 72

hex-dumped storage display  
example 127

hiding the hex-value column 131

High Level Assembler SCLM  
example 306

HistCols option 477

history file

*See also* history file

allocate 249

contents 8

copying an entry 50

deleting an entry 50

initiating interactive reanalysis 50

matching faults 46

moving an entry 50

refreshing 56

selection during real-time  
execution 14

setting name 250, 459

showing duplicate history  
information 50

submitting a batch dump  
reanalysis 50

transmitting an entry 50

viewing a dump data set name 50

viewing the saved fault analysis  
report 50

history file access information  
resetting 56

history file or view

selecting for display 35

history file properties

file menu option 59

history file properties display  
example 51

history file selection

controlling with user exit 402

history file updates

controlling with user exit 403, 404

history file updates pending display  
example 58

history files

PDSE-managed 249

sharing across a sysplex 260

Hogan

sample Formatting user exit 397

HP tag 440

## I

IBM File Manager for z/OS

ISPF data set allocations

required 239

IBM Support Assistant, searching for  
problem resolution 557

identification of column configuration  
source 42

identifying the LE run-time library 244

IDI\_SDUMP\_ACCESS

XFACILIT resource class 229

IDIADATA attributes 314

IDIADATA data sets 507

IDIADATA data sets. 507

IDIADATA DD statement 16, 303

IDIADATA specification via DataSets  
option 458

IDIALLOC REXX command 420

IDIBOPT DDname 454

IDIBOxxx specification via DataSets  
option 458

IDICNF00 452

IDICNF00 parmlib member 453

IDICNFUM 451

IDICNFUM user-options module 452,  
453

IDICNFxx parmlib member 267

alternative data set name 267

IDICZSVC 225

IDIDTEST REXX command 424

IDIDOC

READ access requirement 227

IDIDOC specification via DataSets  
option 458

IDIDOxxx specification via DataSets  
option 458

IDIDSECT concatenation 147

IDIDSECT specification via DataSets  
option 458

IDIDSECTdsn REXX command 424

IDIEventInfo REXX command 425

IDIEXEC DDname 372, 467, 473, 494

IDIEXEC specification via DataSets  
option 458

IDIFREE REXX command 426

IDIGSVRJ sample member 445

IDIHIST data set 516

IDIHIST DD statement 250

IDIHIST specification via DataSets  
option 458

IDIHIST\_GROUP\_DSN

XFACILIT resource class 262

IDIHIST\_USERID\_DSN

XFACILIT resource class 262

IDIHUSRM 363

IDILANGP 315

IDILANGX

invocation parameters 305

messages 556

return codes 533

side file compatibility with Debug  
Tool for z/OS 306

using to create side files 303

IDILANGX attributes 314

IDILANGX data sets 509

IDILANGX DD statement 16, 305

IDILANGX specification via DataSets  
option 458

IDILC attributes 314

IDILC data sets 507, 508

IDILC DD statement 16

IDILC specification via DataSets  
option 458

IDILCOB attributes 314

IDILCOB data sets 508

IDILCOB DD statement 16

IDILCOB specification via DataSets  
option 458

IDILCOBO attributes 314

IDILCOBO data sets 508, 509

IDILCOBO DD statement 16

IDILCOBO specification via DataSets  
option 458

IDILEDS 322

IDILEDS sample member 245

IDILEDS USERMOD 244

IDILPLI attributes 314

IDILPLI data sets 509, 510

IDILPLI DD statement 16

IDILPLI specification via DataSets  
option 458

IDILPLIE data sets 510

IDILPLIE DD statement 16

IDILPLIE specification via DataSets  
option 458

IDIMAPS specification via DataSets  
option 458

IDIModQry REXX command 426

IDINDFUE CICS NoDup(CICSFAST)  
override assembler exit 326

IDIOFF DD statement 368

IDIOPTLM configuration-options  
module 454

IDIOPTLM sample member 454

IDIOPTS DD statement 95

IDIOPTS DDname 451, 452

IDIOPTS options file 328, 454

IDIPANEX sample member 248

IDIPLT 320

IDIPLTD 320

IDIPLTS 320

IDIRegisterFaultEntry REXX  
command 427

IDIREPRT DD statement 16

IDIREPRT DUMMY allocation 17

IDIRFR\_TDUMP\_HLQ

XFACILIT resource class 230

IDIRLOAD DDname

using for CSECT mapping 314

IDIS subsystem for Fault Analyzer 233

IDIS subsystem requirements for  
DB2 237

- IDIS subsystem requirements for Java 237
- IDIS subsystem storage requirements 236
- IDIS\$NDX sample member 9
- IDISBRWS sample member 448
- IDISCICS sample member 322
- IDISCLMA sample member 306
- IDISCLMC sample member 307
- IDISCMDS command table 239
- IDISCMPS sample member 304
- IDISCNF sample member 246
- IDISCNFU sample member 453
- IDISDB2B sample member 349
- IDISDB2X sample member 333
- IDISFEAP sample member 565
- IDISFECL sample member 565
- IDISFEMA sample member 565
- IDISFEMP sample member 565
- IDISFEQP sample member 565
- IDISFESK sample member 565
- IDISFILE sample member 304
- IDISHIST sample member 249
- IDISISPF sample member 239
- IDISJ001 sample member 277
- IDISJ002 sample member 279
- IDISJ003 sample member 281
- IDISJ004 sample member 283
- IDISJ005 sample member 285
- IDISJ006 sample member 291
- IDISJ007 sample member 294
- IDISJ008 sample member 296
- IDISJ009 sample member 300
- IDISJ010 sample member 301
- IDISJ011 sample member 288
- IDISJCTL skeleton member 242
- IDISNAP 18, 515
- IDISNAP input parameter list 19
- IDISPDMA sample member 245
- IDISPLI sample member 245
- IDISPLI USERMOD 245
- IDISPLIA sample member 245
- IDISPLIA USERMOD 245
- IDISRC1 sample member 344
- IDISRFR sample member 246
- IDISROBT sample member 256
- IDISTSOB sample member 258
- IDISUFM1 sample member 392
- IDISUFM2 sample member 395
- IDISUFM3 sample member 435
- IDISUFM4 sample member 397
- IDISUFM5 sample member 401
- IDISUFMX sample member 156
- IDISUSI sample member 219
- IDISUTL1 sample member 417
- IDISVENU sample member 227
- IDISVJPN sample member 227
- IDISVKOR sample member 228
- IDISWRAP 22
- IDISXCUM sample member 247
- IDISXNFY sample member 255, 259
- IDISXPLA sample member 373
- IDISXPLB sample member 373
- IDISXPLC sample member 373
- IDISXPLP sample member 373
- IDISYSDB attributes 314
- IDISYSDB data sets 511
- IDISYSDB DD statement 16
- IDISYSDB specification via DataSets option 458
- IDITABD sample member 245
- IDITABD USERMOD 245
- IDITABX sample member 243
- IDITRACE DD statement 313, 373, 420, 423, 424, 425, 426, 428, 431, 433, 453, 471, 489, 544
- IDITRACE under CICS 324
- IDIUTIL batch utility 353
  - return codes 533
- IDIUTIL batch utility user exit samples 362
- IDIUTIL Delete user exit 413
- IDIUTIL Import user exit 412
- IDIUTIL ListHF user exit 414
- IDIVIEWS DDname 35
- IDIVPASM sample member 341, 345
- IDIVPBLE sample member 345
- IDIVPC sample member 344
- IDIVPCOB sample member 342, 345
- IDIVPDB2 sample member 347
- IDIVPDBB sample member 349
- IDIVPPLE sample member 343
- IDIVPPLI sample member 343, 345
- IDIVSENU
  - READ access requirement 227
- IDIVSJPN
  - READ access requirement 228
- IDIVSKOR
  - READ access requirement 228
- IDIVSxxx specification via DataSets option 458
- IDIWCIDI sample member 447
- IDIWRITE REXX command 428
- IDIWTO REXX command 429
- IDIWTSEL sample member 329
- IDIXCEE 320, 515
- IDIXCEE 515
- IDIXCX52 320, 515
- IDIXCX53 320, 515
- IDIXDCAP 515
- IDIXFA 320, 323
- IDIXFXIT
  - entry specifications 264
  - example 265
  - input parameter list 264
  - return specifications 265
  - user exit purpose 264
- IDIXJAVA 22
- IDIXMAP 320
- IDIXTSEL 515
- IEATDUMP
  - changing the default name for recovery fault recording 246
- IEATDUMP dump title 30
- IEFUSI exit sample 219
- IFAPRDxx parmlib member 367
- IGZIUXB exit
  - locating SYSDEBUG files 311
- implicit refresh 56
- IMPORT control statement 358
- improving Fault Analyzer DB2 performance 333
- IMS
  - Language Environment considerations 335
- IMS and Language Environment 335
- IMS fast duplicate fault suppression 483
- IMS information
  - system-wide information 123
- IMS information display example 125
- Include option 468
- indexing your DSECT data sets (\$DINDEX member) 147
- INFO command 65
- information centers, searching for problem resolution 557
- initiating batch reanalysis 94
- initiating interactive reanalysis 50, 101
- input parameter list for IDISNAP 19
- installation-wide default options 267, 453
- Installing non-ISPF interfaces to access Fault Analyzer history files
  - Fault Analyzer web browser interface 448
- installing the MVS change options/suppress dump exit
  - IDIXDCAP 243
- installing the MVS post-dump exit
  - IDIXTSEL 329
- interactive displays
  - copying to a file 75
- interactive options line 98
- interactive reanalysis 7
  - data sets used 160
  - initiating 50, 101
  - options 97
  - performing 97
- interactive reanalysis options
  - fault entry list display options menu option 60
- interactive reanalysis options display example 97
- interactive reanalysis report display example 103
- interactive reanalysis status display example 102
- interactive reanalysis under CICS 447
- interactive report
  - abend job information 129
  - detailed event information 107
  - displaying associated storage areas 130
  - displaying source code 135
  - displaying storage locations 138
  - event summary 105
  - expanding messages and abend codes 134
  - Fault Analyzer options 129
  - general information 102
  - synopsis 105
  - user notes 129
- InteractiveExitPromptSeconds
  - option 478
- Internet
  - searching for problem resolution 557
- introduction 3
- invocation for CICS transaction abends 221

- invocation for non-CICS transaction
  - abends 219
- Invoking Fault Analyzer from Java catch block 22
- Invoking Fault Analyzer from Java dump events 24
- invoking Fault Analyzer from PL/I PLIDUMP 245
- invoking Fault Analyzer from SDSF 241
- invoking the ISPF interface 32
- invoking the LOOKUP command using cursor selection 241
- IPL requirement 225
- ISPF interface 31
  - commands 61
  - customizing 271
  - invoking 32
  - on-line help 32
  - providing defaults for new users 242
  - security considerations 88
- ISPF packed data format
  - using with Fault Analyzer 316
- ISPF selection panel update 240
- ISPF split screen support 32
- ISPLIBD 239
- ISRDDN 239
- ISRFIND 239
- IVP testing
  - CICS 345

## J

- Java
  - IDIS subsystem requirements 237
  - invoking Fault Analyzer from 22
- Java abend analysis
  - selecting dump data set 175
- Java analysis 175
  - displaying the information in the interactive report 177
  - setting options 175
- Java application abends
  - required LE options 223
- Java dump analysis
  - creating history file fault entry 175
- Java event details display example 180
- Java event summary display example 179
- Java fault entry reanalysis 176
- Java information
  - system-wide information 127
- Java information display example 182
- Java interactive reanalysis report display example 178

## K

- keywords, syntax diagrams xi
- knowledge bases, searching for problem resolution 557

## L

- L tag 440
- Language Environment
  - DB2 considerations 331

- Language Environment (*continued*)
  - IMS considerations 335
  - not in LINKLIST 244
- Language Environment abnormal termination exit
  - enabling 243
- Language Environment abnormal termination exit for CICS
  - enabling 320
- Language Environment heap analysis
  - system-wide information 127
- Language Environment heap analysis information display example 128
- Language Environment options required for invocation of Fault Analyzer 222
- Language option 479
- last accessed history file entries
  - file menu option 60
- last accessed history file entries display example 39
- last accessed history files or views
  - file menu option 60
- last accessed history files or views display example 38
- last CICS 3270 screen buffer
  - system-wide information 113
- last CICS 3270 screen buffer display example 113
- last CICS 3270 screen buffer hex
  - system-wide information 113
- last CICS 3270 screen buffer hex display example 114
- LE abnormal termination exit
  - using with MVS change options/suppress dump exit 221
- LE options 222, 328
- LE options required for CICS abends 223
- LE options required for non-CICS abends 222
- LE options required to capture Java application abends 223
- level 88 items display example 134
- LI tag 441
- library names after you finish
  - installing 216
- license inquiry 569
- limiting the size of minidumps 271
- LINKLIST
  - modules in 225
- List REXX command 430
- list user notes
  - services menu option 60
- list views
  - file menu option 60
- LISTHF control statement 354
- listing/side file mismatch display example 159
- listing/side file trace display example 160
- listings
  - pointing to 271
  - storing 303
- LOADER restriction 13
- locale name 510
- Locale option 480

- locating compiler listings or side files 311
- lock flag 58, 80, 517
- locking of fault entries to prevent deletion 58
- LOOKC command 66, 241
- lookup
  - services menu option 60
- LOOKUP command 66
- lookup search and browse display example 74
- LoopProtection option 480
- LPA
  - modules in 225, 226
  - placing modules in 365
- LPA module compatibility 209
- LST data area 520

## M

- main report sections 185
- maintaining Fault Analyzer 365
- making Fault Analyzer modules available 225
- managing history file fault entry
  - access 261
- managing history files 353
- managing history files across MVS systems without shared DASD 253
- managing recovery fault recording data set access 228
- mapping storage areas using DSECT information 145
- MATCH ALL match condition 49
- MATCH command 46, 66
- MATCH CSR match condition 49
- matching fault entries 46
- matching fault history entries
  - using a wildcard 49
- maximizing CICS transaction abend analysis performance 329
- maximum call depth 187
- MaxMinidumpPages option 481
- menus
  - action-bar pull-down 58
- message and abend code explanation repository
  - setting up 227
- Message and Abend Code Explanation user exit 386
- message explanation display example 135
- message id look-up display example 74
- messages 535
  - Fault Analyzer 535
  - IDILANGX 556
  - search order 364
  - system-wide information 120
  - user-defined 363
- migrating from an earlier version of Fault Analyzer 205
- migrating from V10.1 to V11.1 205
- migrating from V11.1 to V12.1 205
- migrating from V6.1 to V7.1 207
- migrating from V7.1 to V8.1 206
- migrating from V8.1 to V9.1 206
- migrating from V9.1 to V10.1 205

- minidump
  - concept 3
  - limiting size 271
- minimum storage requirements 218
- modifying your ISPF environment 239
- moving a history file entry 50, 87
- moving a range of fault history entries 87
- MTRACE records
  - system-wide information 128
- MTRACE Records display example 128
- multicultural support 337, 339, 479, 480
- MVS change options/suppress dump exit using with LE abnormal termination exit 221
- MVS dump data set size 224

## N

- naming CSECTs for Fault Analyzer 311
- national language
  - setting 337, 339
- new history file allocation
  - file menu option 60
- new history file allocation display example 53
- NEXT command 67
- NFY data area 522
- NODUMP
  - using with EXEC CICS ABEND command 322
- NODUMP keyword on EXEC CICS ABEND command 322
- NoDup option 482
- non-ISPF interfaces to access Fault Analyzer history files
  - installing 445
  - using 193
- non-REXX user exit buffered data format 503
- not found dump status 82
- Note REXX command 432
- NOTEL tag 441
- NOTELIST command 67
- Notification user exit 405
- Notification user exit (dump registration) 411

## O

- obtaining load modules from CA-Panexec 247
- ON ERROR
  - consideration when using 224
- open files
  - system-wide information 110
- options
  - \_IDI\_OPTS environment variable 455
  - \_IDI\_OPTSFILE environment variable 454
  - AdditionalIDIOffDD 455
  - batch reanalysis 89
  - CICSDumpTableExclude 456
  - CICSTraceMax 456
  - configuration-options module IDIOPTLM 454

- options (*continued*)
  - cumulative 452
  - DataSets 271, 458
  - DeferredReport 463
  - Detail 270, 465
  - DumpDSN 466
  - DumpRegistrationExits 466
  - ErrorHandler | NoErrorHandler 467
  - Exclude 269, 468
  - Exits 473
  - FaultID 475
  - general description 451
  - GenerateSavedReport 476
  - HistCols 477
  - IDICNF00 parmlib member 453
  - IDIOPTS 454
  - Include 269, 468
  - installation-wide 453
  - interactive reanalysis 97
  - InteractiveExitPromptSeconds 478
  - Language 337, 339, 479
  - Locale 480
  - LoopProtection 480
  - MaxMinidumpPages 271, 481
  - NoDup 482
  - PARM field 455
  - PermitLangx 490
  - PreferredFormattingWidth 491
  - PrintInactiveCOBOL |
    - NoPrintInactiveCOBOL 492
  - purpose 451
  - Quiet 271
  - Quiet | NoQuiet 492
  - RDZClient 493
  - RefreshExits 493
  - RetainCICSDump 495
  - RetainDump 495
  - setting 337, 339
  - Source | NoSource 496
  - SpinIDIREPRT |
    - NoSpinIDIREPRT 497
  - StoragePrintLimit 497
  - StorageRange 498
  - syntax rules 452
  - SystemWidePreferred 499
  - UseIDISTime 501
  - user options file 454
  - user-options module IDICNFUM 453
  - WDZClient 493
  - where set 451
  - where to specify 452
- options in effect 170
- options in effect display example 171
- options menu
  - deletion options 77
- organization of this book xi
- OS/VS COBOL
  - compiling programs for 284
- over-typing existing values 48

## P

- P tag 441
- packed data format, ISPF
  - using with Fault Analyzer 316
- PARM field options 455
- PDSE-managed history files 249

- performing CICS IVP testing 345
- performing interactive reanalysis under CICS 194
- performing Java analysis 175
- PermitLangx option 490
- PF keys
  - showing 33
- PL/I for MVS and VM
  - compiling programs for 294
- PL/I ON ERROR
  - consideration when using 224
- PL/I version 2 release 3
  - USERMOD to facilitate invocation of Fault Analyzer 245
- PLIDUMP
  - always invoking Fault Analyzer from 245
- point of failure 519
- point-and-shoot fields 103
- pointing to listings 271
- pointing to REXX exec libraries 271
- preferred formatting width
  - setting 72
  - view menu option 61
- preferred formatting width display example 73
- PreferredFormattingWidth option 491
- Preparing programs for IBM Problem Determination Tools 273
- preparing to customize Fault Analyzer 213
- PREV command 67
- preventing LE from causing the CICS trace to wrap 328
- PrintInactiveCOBOL |
  - NoPrintInactiveCOBOL option 492
- problem determination
  - describing problems 562
  - determining business impact 561
  - submitting problems 562
- production environment 275
- program SNAP interface 18
- prompting for compiler listing or side file 156
- providing compiler listings or Fault Analyzer side files 303
- providing explanations for application-specific messages 363
- providing ISPF interface defaults for new users 242
- providing the name of a history file to Fault Analyzer 250
- pull-down menus 58

## Q

- Quiet | NoQuiet option 492
- QUIT command 68

## R

- range delete
  - See also* deleting many fault history entries
  - grouping entries 46
- RDZClient option 493



- READ access requirement
  - IDIDOC 227
  - IDIVSENU 227
  - IDIVSJPN 228
  - IDIVSKOR 228
- real-timeabend analysis 3
- real-time analysis 13
  - controlling with options 15
- real-time analysis report 16
- real-time fault analysis report
  - viewing 70
- real-time reports
  - combining 17
  - controlling SYSOUT class 17
  - suppressing 17
- real-time SNAP analysis 5
- reanalysis
  - batch 50, 89
  - interactive 97
- reanalyzing a fault 31
- recovery fault recording
  - changing the default IEATDUMP data set name 246
  - Fault Analyzer IDIS subsystem usage 233
  - overview 28
  - verifying the set-up 351
- refresh
  - view menu option 61
- REFRESH command 56, 68
- refresh processing 161
- RefreshExits option 493
- refreshing cached messages 364
- refreshing the fault entry list display 56
- region size requirements 218
- remove blank lines
  - view menu option 61
- remove help text
  - view menu option 61
- remove pseudo assembler instructions
  - view menu option 61
- repeatable items, syntax diagrams xi
- report detail
  - controlling 270
- report examples 191
- report sections
  - abend job information 189
  - epilog 190
  - event details 188
  - main 185
  - options 190
  - prolog 186
  - summary 186
  - synopsis 186
  - system-wide information 189
- report user exit
  - See Analysis Control user exit
- RESET command 68
- resetting history file access
  - information 56
- restricting change of history file
  - settings 226
- RetainCICSDump option 495
- RetainDump option 495
- return codes from batch reanalysis 533
- return codes from IDILANGX 533

- return codes from IDIUTIL batch utility 533
- REXX commands 418
  - Evaluate 418
  - IDIALLOC 420
  - IDIDDTTEST 424
  - IDIDSECTdsn 424
  - IDIEventInfo 425
  - IDIFREE 426
  - IDIModQry 426
  - IDIRegisterFaultEntry 427
  - IDIWRITE 428
  - IDIWTO 429
  - List 430
  - Note 432
- REXX exec libraries
  - pointing to 271
- REXX SAY instruction 377
- REXX TRACE instruction 377
- RFR dump titles 30
- RFR TDUMP XFACILIT example 231
- routing codes
  - used by Fault Analyzer WTO messages 12
- row count 34, 106
- RPTFIND command 68
- RUNCHAIN command 69
- running Fault Analyzer with similar third-party products 223

## S

- sample CICS definition job 322
- sample customized ISPF interface
  - front-end 565
- sample data set members
  - IDICNFxx 268
  - IDIGSVRJ 445
  - IDILEDs 245
  - IDIOPTLM 454
  - IDIPANEX 248
  - IDIS\$NDX 9
  - IDISBRWS 448
  - IDISCICS 322
  - IDISCLMA 306
  - IDISCLMC 307
  - IDISCMPS 304
  - IDISCNF 246
  - IDISCNFU 453
  - IDISDB2B 349
  - IDISDB2X 333
  - IDISFEAP 565
  - IDISFECL 565
  - IDISFEMA 565
  - IDISFEMP 565
  - IDISFEQP 565
  - IDISFESK 565
  - IDISFILE 304
  - IDISHIST 249
  - IDISISPF 239
  - IDISJ001 277
  - IDISJ002 279
  - IDISJ003 281
  - IDISJ004 283
  - IDISJ005 285
  - IDISJ006 291
  - IDISJ007 294

- sample data set members *(continued)*
  - IDISJ008 296
  - IDISJ009 300
  - IDISJ010 301
  - IDISJ011 288
  - IDISPDM 245
  - IDISPLI 245
  - IDISPLIA 245
  - IDISRC1 344
  - IDISRFR 246, 247
  - IDISROBT 256
  - IDISTSOB 258
  - IDISUFM1 392
  - IDISUFM2 395
  - IDISUFM3 435
  - IDISUFM4 397
  - IDISUFM5 401
  - IDISUFMX 156
  - IDISUSI 219
  - IDISUTL1 417
  - IDISVENU 227
  - IDISVJPN 227
  - IDISVKOR 228
  - IDISXNFY 255
  - IDITABD 245
  - IDITABX 243
  - IDIVPASM 341, 345
  - IDIVPBLE 345
  - IDIVPC 344
  - IDIVPCOB 342, 345
  - IDIVPDB2 347
  - IDIVPDBB 349
  - IDIVPPLE 343
  - IDIVPPLI 343, 345
  - IDIWCIDI 447
  - IDIWTSEL 329
  - IDIXMIT 259
- Sample data set members
  - IDISXPLA 373
  - IDISXPLB 373
  - IDISXPLC 373
  - IDISXPLP 373
- sample reports 191
- sample XFACILIT implementation 263
- saved report
  - viewing 70
- saved report display example 71
- SAY
  - REXX instruction 377
- SCLM
  - including an IDILANGX step in your translator 306
- SDUMP recovery fault recording data sets 229
- SDUMP SVC
  - screening 321
- security considerations 88
- selecting a CICS auxiliary trace data set 173
- selecting an address space to analyze 165
- selecting fault entries 46
- selecting the CICS dump data set 163
- selecting the Java dump data set 175
- sending fault entries from one system to another 258
- SETFAULTPREFIX control statement 356

- SETMAXFAULTENTRIES control statement 357
- SETPROG command
  - managing modules using 365
- setting and changing default options for the site 267
- setting deletion options 77
- setting options for CICS system abend analysis 163
- setting options for Java analysis 175
- setting preferred formatting width 72
- setting the name of the history file 250
- setting the national language 337, 339
- setting up existing programs for fault analysis 10
- setting up history files 249
- setting up the message and abend code explanation repository 227
- setting up views 250
- SFA command 241
- sharing of history files across a sysplex 260
- sharing of history files across a sysplex with mixed levels of Fault Analyzer 209
- SHOW command 69
- showing duplicate history information 50
- showing PF keys 33
- side file formatting utility 315
- side files
  - advantage over listings 10
  - attributes 313
  - naming 310
  - storing 303
- SLIP TRAP
  - effect on real-time analysis 220
- SMF type 89 record 28, 367
- Software Configuration & Library Manager
  - including an IDILANGX step in your translator 306
- Software Support
  - contacting 560
  - describing problems 562
  - determining business impact 561
  - receiving updates 559
  - submitting problems 562
- Sorting and matching fault entries 45
- Sorting and matching table displays 169
- sorting fault entries 46
- Source | NoSource option 496
- source information files 273
- source support 273
- special history file data set members 9
  - \$\$BACKUP 9
  - \$\$INDEX 9
- special processing of Language Environment CEEWUCHA abends 11
- specify compiler listing or side file display example 158
- specify move/copy options display example 86
- specify XMIT options display example 88
- specifying a default column layout 252
- specifying an alternative parmlib data set for IDICNF00 246
- specifying an initial fault entry selection criteria 252
- Specifying CICS Trace Selection Parameters 174
- specifying the cultural environment 271
- SpinIDIREPT | NoSpinIDIREPT option 497
- SQLCA 123, 500
- SSRANGE
  - CICS COBOL 222
- starting the Fault Analyzer IDIS subsystem 235
- status pop-up display 101
- STCK command 70
- STCK conversion display example 153
- stopping the Fault Analyzer IDIS subsystem 238
- storage areas
  - system-wide information 126
- Storage Disassemble display example 151
- storage DSECT mapping entry display example 146
- storage DSECT mapping map display example 147
- storage recommendations 218
- storage requirements 218
  - CICS 329
- storage RUNCHAIN command entry display example 149
- StoragePrintLimit option 497
- StorageRange option 498
- stored procedures
  - DB2 331
- storing listings or side files 303
- submitting a batch dump reanalysis 50
- substitution symbols in data set names 378, 461
- subsystem for Fault Analyzer IDIS 233
  - user exits running from 467
- successful analysis criteria 519
- summarized CICS trace
  - system-wide information 114
- summarized CICS trace display example 115
- summary of exit usage 221
- supported application environments 9
- suppressed copybooks 185
- suppressed dump status 82
- suppressing noncritical syslog messages 271
- suppressing real-time reports 17
- SVC 51
  - screening 321
- SVC dump exit 329
- SVC dump registration exit 222
- SVC dump screening 321
- SVC installation
  - using IDICZSVC to perform dynamically 225
- SVCDUMP dump title 30
- symbols in data set names 378, 461
- synopsis 105, 167
- synopsis display example 105
- syntax diagrams, how to read xi
- SYSADATA DD statement 303
- SYSCLONE MVS system symbol
  - used as IDICNFxx parmlib member suffix 267
- SYSDEBUG files
  - locating using the EQAUEDAT exit 311
  - locating using the IGZIUXB exit 311
- syslog messages
  - suppressing 271
- SYSLOG summary 18
- SYSMDUMP
  - ASA printer control characters 466
  - logical record length 466
  - specifying as input to batch reanalysis 466
- SYSMDUMP open prompt 99
- SYSOUT class of real-time reports
  - controlling 17
- sysplex
  - sharing of history files 260
- Sysplex-wide subsystem
  - inter-communication 234
- system dump suppression
  - controlling with user exit 403
- system-wide information
  - CICS information 112
    - CICS levels, commareas, and channels 117
    - CICS trace formatting 115
    - last CICS 3270 screen buffer 113
    - last CICS 3270 screen buffer hex 113
    - summarized CICS trace 114
  - DB2 information 120
  - IMS information 123
  - Java Information 127
  - Language Environment heap analysis 127
  - messages 120
  - MTRACE records 128
  - open files 110
  - storage areas 126
  - WebSphere Information 127
- system-wide messages display example 120
- system-wide open files display example 110
- system-wide storage areas display example 126
- SystemWidePreferred option 499
- SYSUDUMP SYSOUT class
  - allocation of IDIREPT 16

## T

- TACB
  - fields used for duplicate determination under CICS 487
- TDUMP recovery fault recording data sets 229
- temporarily deinstalling Fault Analyzer 367

- temporary data set
  - used for compiler listing or side files 314
- test environment 275
- TEST option considerations 309
- TEST(NONE,SYM,SEPARATE) COBOL compiler option 309, 311
  - restriction with display of pseudo assembler instructions 137
- TEST(STMT,SYM,NOHOOK,SEPARATE) Enterprise PL/I compiler option 309, 311
- TH tag 442
- third-party products and Fault Analyzer 223
- title 567
- total rows displayed 34, 106
- TRACE
  - REXX instruction 377
- tracing
  - CICS 322
  - user exits 373
- transaction abend control block (TACB)
  - fields used for duplicate determination under CICS 487
- transmitting a history file entry 50, 87
- transmitting a range of fault history entries 87
- TSO
  - REXX environment restrictions 372
- turning off Fault Analyzer using an environment variable (\_IDI\_OFF) 368
- turning off Fault Analyzer using the IFAPRDxx parmlib member 367
- turning off Fault Analyzer with a JCL switch (IDIOFF) 368

## U

- U tag 442
- UFM data area 523
- UL tag 443
- update the ISPF selection panel 240
- updates pending 57
- UseIDISTime option 501
- user exit type specific data area 372
- user exits 369
  - Analysis Control 377
  - Analysis Control (dump registration) 380
  - Compiler Listing Read 382
  - data area version checking 373
  - End Processing 402
  - End Processing (fault entry refresh) 404
  - Formatting 390
  - IDIUTIL Delete 413
  - IDIUTIL Import 412
  - IDIUTIL ListHF 414
  - invocation parameters 372
  - Message and Abend Code Explanation 386
  - Notification 405
  - Notification (dump registration) 411
  - supported programming languages 372
  - tracing 373

- user exits (*continued*)
  - used with CICS system abend analysis 163
  - using substitution symbols in data set names returned 378
- user fields update prompt example 83
- user modifications
  - parmlib data set name 267
- user note list display example 144
- user notes 129
  - creating and managing 140
- user notes update prompt example 145
- user options file 454
- user-defined messages 363
- user-options file 451
- user-options module 451
- user-options module IDICNFUM 453
- user-selected 73
- user-selected message or abend code explanations
  - displaying 73
- user-specific report formatting (EXEC command) 154
- USERMODs
  - IDILEDS 244
  - IDISPLI 245
  - IDISPLIA 245
  - IDITABD 245
  - IDITABX 243
  - IDIWTSSEL 329
  - restore and re-apply 365
- using CFA to FORCEPURGE the currently analyzed task 321
- using IDIOPTS DDname with CICS 328
- using non-ISPF interfaces to access Fault Analyzer history files
  - using the Fault Analyzer client for IBM Rational Developer for System z 193
- Using SLIP,COMP=0C4 with Fault Analyzer 30
- using the Fault Analyzer IDIS subsystem 233
- using the Fault Analyzer web browser interface 194
- using the MATCH command 48
- using the XFACILIT resource class for history file fault entries 261
- using the XFACILIT resource class for SDUMP RFR data sets 229
- using the XFACILIT resource class for TDUMP RFR data sets 230
- using views 35
- USRHDLR(CEEWUCHA)
  - special processing 11
- UTL data area 529

## V

- variables, syntax diagrams xi
- verifying the customization of Fault Analyzer 341
- verifying the customization of Fault Analyzer under CICS 345
- verifying the IDITABD USERMOD installation 345

- verifying the IDIXCEE Language Environment exit enablement 345
- verifying the recovery fault recording set-up 351
- verifying the use of Fault Analyzer through ISPF 351
- verifying the use of Fault Analyzer with assembler 341
- verifying the use of Fault Analyzer with C 344
- verifying the use of Fault Analyzer with COBOL 342
- verifying the use of Fault Analyzer with DB2 347
- verifying the use of Fault Analyzer with PL/I 343
- view considerations if not using XFACILIT resource class 253
- view list display example 40
- viewing a dump data set name 50
- viewing a fault analysis report 50
- viewing a real-time analysis report 70
- viewing a saved report 70
- viewing fault entry duplicate history 83
- viewing fault entry information 78
- views
  - setting up 250
  - using 35
- VIEWS command 70
- VS COBOL II
  - compiling programs for 282

## W

- WDZClient option 493
- web browser interface 194
- WebSphere information
  - system-wide information 127
- WebSphere or Java dump analysis 175
- wildcard matching 49
- working with applications that use a non-Language Environment run time 244
- WTO routing and descriptor codes used by Fault Analyzer 12

## X

- XCF 234, 261
- XFACILIT implementation example 263
- XFACILIT resource class
  - using to manage history file fault entries 261
  - using to manage SDUMP recovery fault recording data sets 229
  - using to manage TDUMP RFR data sets 230
- XFACILIT resource classes used by Fault Analyzer
  - IDI\_SDUMP\_ACCESS 229
  - IDIHIST\_GROUP\_DSN 262
  - IDIHIST\_USERID\_DSN 262
  - IDIRFR\_TDUMP\_HLQ 230
- XPL data area 530

## **Z**

z/OS XL C and C++  
compiling programs for 296







Product Number: 5655-W69

Printed in USA

SC19-3671-00

